1993年8月1日発行（毎月1回1日発行）第12巻8号通巻136号 昭和58年11月2日第三種郵便物認可

PERSONAL COMPUTER MAGAZINE for MZ, X1, and X68000

Oh!X

8
1993

特集 C言語実践的入門

特別企画 夏真っ盛り，アマチュアリズムのX68000

新製品紹介Easydraw SX-68K/試用レポートRED ZONE

ASK3アクセサリプログラミング/大人のためのX68000

SOFT BANK オー／エックス
定価600円

SHARP
目の付けどころが、
シャープでしょ
夢の、頂きへ。
68ワールドの最高峰。
X68030
32bit PERSONAL WORKSTATION

32ビットパーソナルワークステーション

演算速度4.3倍(当社10MHz機比)/2.4倍(当社XVI比)※1、動画ウィンドウに見る新創造次元。選ばれた人だけが持つ感性によってX68030の扉はひらかれる。

X68000シリーズとして初の32ビットMPU MC68EC030を搭載して高速化を実現。

データキャッシュ、プログラムキャッシュをそれぞれ256バイト搭載したクロック周波数25MHzの高速32ビットMPUを搭載。演算速度は2倍以上(当社従来比)※1の高速化を実現しました。また数値演算プロセッサMC68882※2(25MHz)もサポート。大量の実数演算を必要とするクリエイティブワークやGUI環境の操作性など、実行速度の飛躍的な向上が図られています。(当社従来比)

※1 Dhrystn(四則演算)比。25MHz・データキャッシュオン・プログラムキャッシュオンでMC68000/10MHz時の約4.3倍、16MHz時の約2.4倍。
※2 数値演算プロセッサCZ-5MP1標準価格54,800円(税別):本体内の専用ソケットに取りつけ可能。

65,536色表示、動画表示を実現。さらにパワーアップしたSX-WINDOWver.3.0。

X68000独自の本格的ウィンドウシステムとして定評の「SX-WINDOWver.2.0」をさらに強化した「SX-WINDOWver.3.0」を標準装備。新たに、65,536色の自然色グラフィック表示を可能とした『グラフィックウィンドウ』※を搭載。またアニメーション動画をウィンドウ上で表現でき、手軽にコンピュータアニメーションが楽しめる『CGAウィンドウ』、さらに従来のエディタのイメージを一新、高度な日本語文書作成をサポートするSX-WINDOW対応の高機能日本語マルチフォントエディタを標準装備。アウトラインフォントの展開もさらに高速化が図られています。

※SX-WINDOW上の512×512ドットのエリア内で表示可能。

GUIに対応する大容量メインメモリを搭載。

メインメモリは標準で4Mバイト、複数のアプリケーションをウィンドウ上で同時に使用するなど大量のデータ処理に対応。また本体内の増設で、I/Oスロットを使用せず最大12Mバイトまで拡張できます。拡張したメモリはすべて32ビットバスによる高速アクセスが可能、優れた拡張環境でシステムパワーアップをサポートします。

※メモリ増設には、4MB内部増設RAMボードCZ-5BE4標準価格54,800円(税別)、4MB増設RAMモジュールCZ-5ME4標準価格49,800円(税別)をご使用ください。なおCZ-5ME4はCZ-5BE4上に装着します。

X68000シリーズの高機能を継承した上で、さらに使いやすさの向上を図ったコンパチビリティ重視設計※1、すぐに使える高機能ソフトを標準装備。

●25MHzでは速すぎるアプリケーションも、従来のクロック周波数(10MHz/16MHz)で動作可能なソフトコンパチ重視設計●65,536色同時発色の自然色グラフィックス(最大表示エリア512×512ドット)、1024×1024ドットの実画面エリアを持つ高解像度表示能力(最大表示エリア768×512ドット・カラー液晶ディスプレイ使用時※2は640×480ドット)、疑似高解像度スーパーインポーズ(インターレース方式/512×512ドット・専用ディスプレイテレビ使用時)を装備した高精細度自然色グラフィックス機能。●外部MIDI音源もコントロール可能※3、ウィンドウ上で手軽にコンピュータミュージックが楽しめるMIDI音源対応デバイスドライバ搭載●ステレオ8オクターブ8重和音FM音源、ADPCM搭載●プリンタ、RS-232C、SCSI、オーディオ入出力、イメージ入力など多彩なインターフェイスを装備。●日本語変換効率や操作性を高めた日本語フロントプロセッサASKver3.0搭載。●従来のエディタのイメージを一新したSX-WINDOW対応の高速多機能日本語マルチフォントエディタ標準装備●日本語マルチフォントエディタ中に貼り付ける絵やグラフなどが簡単に作成できるグラフィックパターンエディタ●MIDI対応のX-BASIC。

※1 アプリケーションソフトおよび周辺機器のうち、一部動作しないものがあります。詳しくはシャープお客様相談窓口にお問い合わせください。
※2 10.4型カラー液晶ディスプレイLC-10C1-H標準価格598,000円(税別)、接続ケーブルAN-1515X標準価格4,200円(税別)をご使用ください(SX-WINDOW対応アプリケーションのみ、色数に制限があります)。
※3 別売のMIDIインターフェイスが必要です。

5.25"FDDマンハッタンシェイプシリーズ

■X68000伝統のマンハッタンシェイプを継承 ■5.25インチFDD2基搭載 ■80MBハードディスク内蔵(CZ-510C)※ ■マウス・トラックボール標準装備 ■ASCII準拠フルキーボード採用
※CZ-500Cには、2.5インチ80MB内蔵用ハードディスクドライブCZ-5H08/2.5インチ160MB内蔵用ハードディスクドライブCZ-5H16を用意しています。

X68030
32bit PERSONAL WORKSTATION

本体+キーボード+マウス・トラックボール
5.25インチFDDタイプ CZ-500C-B(チタンブラック)標準価格398,000円(税別)
HDタイプ CZ-510C-B(チタンブラック)標準価格488,000円(税別)
14型カラーディスプレイ
CZ-608D-B(チタンブラック)標準価格94,800円(税別・チルトスタンド同梱)

3.5"FDDコンパクトシリーズ

■32ビットのハイパワーを凝縮したコンパクトフォルム ■2DD対応3.5インチFDD2基搭載 ■80MBハードディスク内蔵(CZ-310C)※ ■マウス標準装備 ■コンパクトキーボード採用
※CZ-300Cには、2.5インチ80MB内蔵用ハードディスクドライブCZ-5H08/2.5インチ160MB内蔵用ハードディスクドライブCZ-5H16を用意しています。

Compact

本体+キーボード+マウス
3.5インチFDDタイプ CZ-300C-B(チタンブラック)標準価格388,000円(税別)
HDタイプ CZ-310C-B(チタンブラック)標準価格478,000円(税別)
14型カラーディスプレイ
CZ-608D-B(チタンブラック)標準価格94,800円(税別・チルトスタンド同梱)

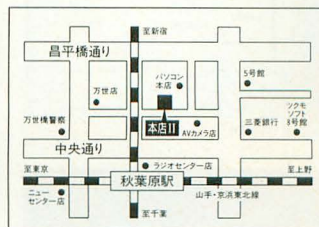
わんさかフェア with X68

日時

●7月24日(土) 10:45〜19:30
●7月25日(日) 10:15〜19:00

場所

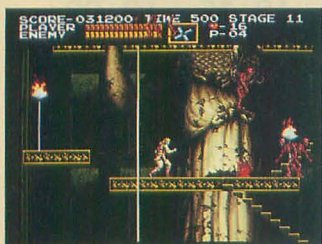
ツクモパソコン本店II 3階

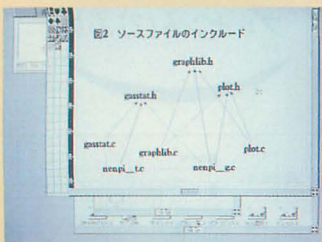
話題のX68030はもちろん最新のソフトや周辺機器を体験できる。さらに激安の目玉商品がわんさか。来場の方には謎のプレゼント(?)あり

●お問い合わせ先●
ツクモパソコン本店II 3階
☎03-3253-1899

■お問い合わせは… シャープ株式会社 コンシューマーセンター 西日本相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

資料請求券 X68030 Oh!X 8係

特別企画　アマチュアリズムのX68000

THE USER'S WORKS

悪魔城ドラキュラ

餓狼伝説

Easydraw SX-68K

DōGA CGアニメーション講座

Oh!X

C O N T

〈スタッフ〉
●編集長／前田 徹 ●副編集長／植木章夫 ●編集／浅井研二 山田純二 豊浦史子 ●協力／有田隆也 中森 章 林 一樹 吉田幸一 華門真人 吉田賢司 朝倉祐二 大和 哲 村田敏幸 丹 明彦 三沢和彦 長沢淳博 司馬 護 石上達也 柴田 淳 瀧 康史 横内威至 進藤慶到 ●カメラ／杉山和美 ●イラスト／山田晴久 江口響子 高橋哲史 川原由唯 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 ADGREEN ●校正／グループごじら

表紙絵：須藤　牧人

1993 AUG. 8

E N T S

UNIXはAT & T BELL LABORATORIESのOS名です。
Machはカーネギーメロン大学のOS名です。
CP/M, P-CPM, CP/Mupls, CP/M-86 CP/M-68K, CP/M-8000, DR-DOSはデジタルリサーチ
OS/2はIBM
MS-DOS, MS-OS/2, XENIX, MACRO80, MS C, WindowsはMICROSOFT
MSX-DOSはアスキー
OS-9, OS-9/68000, OS-9000, MW CはMICROWARE
UCSD p-systemはカリフォルニア大学理事会
TURBO PASCAL, TURBO C, SIDEKICKはBORLAND INTERNATIONAL
LSI CはLSI JAPAN
HuBASICはハドソンソフト
の商標です。その他，プログラム名，CPU名は一般に各メーカーの登録商標です。本文中では"TM", "R"マークは明記していません。
本誌に掲載されたプログラムの著作権はプログラム作成者に保留されています。著作権上，PDSと明記されたもの以外，個人で使用するほかの無断複製は禁じられています。

■広告目次

先が、面白くなる。

ウィンドウ環境のプラットホームを確立、SX-WINDOW ver.3.0

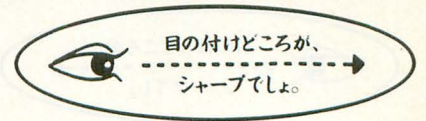
■実画面：1,024×1,024ドット、表示画面：768×512ドット

●この画面は広告用に作成した、機能を説明するためのイメージ画面です。また、各種アイコン等は、SX-WINDOW ver.3.0がもつ機能を使って作成したもので、標準装備のものとは異なるものもあります。
●本広告中のエディタで表示している文字のフォントはZeit社の、「書体倶楽部」のフォントを使用しています。

シャープ株式会社
●お問い合わせは…コンシューマーセンター西日本相談室　〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
電子機器事業本部システム機器営業部　〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

資料請求券 ペリフェラル Oh!X 8体

SHARP

目の付けどころが、シャープでしょ。

こ見たGUIの新展開。

❶マルチフォントエディタ編集例。文字ごとに文字種、文字の大きさの指定、修飾が可能で、イメージデータの貼り付けもOK。

❷CONFIG.SYSやAUTOEXEC.BATなどの編集に便利な「エディタ」モードの例。このように日本語マルチフォントエディタは、用途に合わせてカスタマイズできます。

❸❶の画面をプリンタで印字した例。対応プリンタも増えました。

❹「パターンエディタ」で作成したデータを、背景に設定できます。

❺バージョンアップした日本語フロントプロセッサASK68K ver.3.0の辞書メンテナンスがウィンドウ上で可能。

❻オリジナルに作成したアイコンパターンの例。

❼アイコンデータや背景データを作成する「パターンエディタ」。文字の貼り付けなど、編集機能も一段とフレンドリーに。

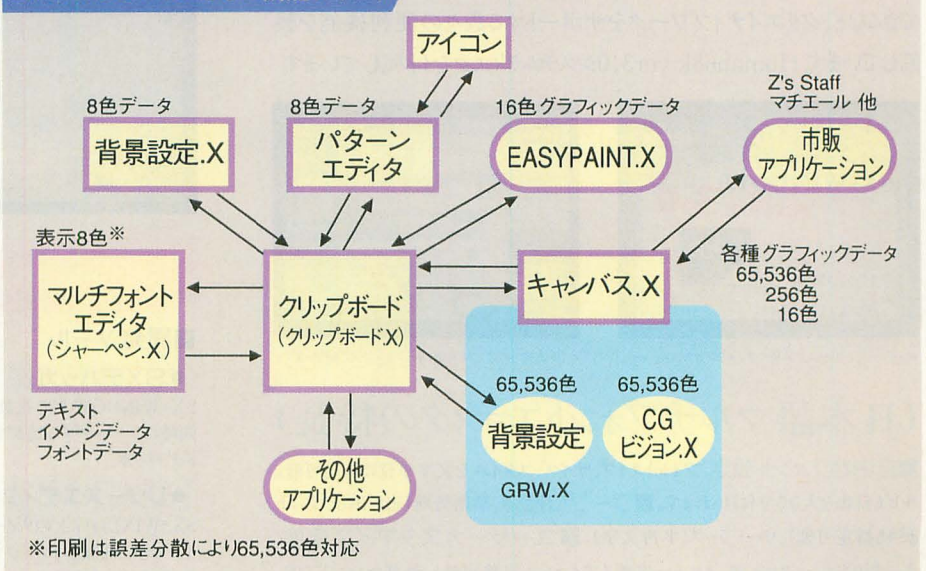
❽512×512ドットの範囲内で65,536色の表示が可能。

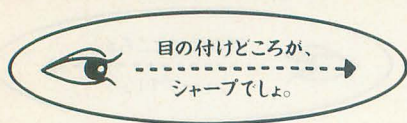
❾さまざまなグラフィックフォーマットに対応しています。

❿任意のサイズに縮小・拡大表示可能。

⓫異なる画像フォーマットへのコンバートができます。

⓬「CGAウィンドウ」、65,536色(最大)のコンピュータアニメーション表示が可能です。

発展性のあるプラットホームとしてのウィンドウシステム、
SX-WINDOWver.3.0が提供する新たなGUI環境が
本格的なウィンドウ時代を予見する――――。
国産オリジナルウィンドウとしての意味、未来への確かなビジョン、
ユーザーインタフェースや高速化へのゆるぎない探求が
ここに凝縮されています。
65,536色表示はもちろん、さまざまな画像フォーマット対応、
イメージデータのコピー&ペースト、
動画、音楽/音声再生をサポートするマルチメディア環境。
そして、何よりもこれらが密接に連携して
統合的にハンドリングできるエキサイティングな環境を創造しています。
未来を照準に入れたウィンドウアーキテクチャ、
そのインテリジェンスがいよいよX68030/X68000シリーズで享受できます。

SHARP

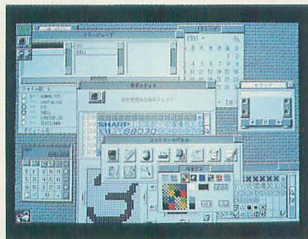
X68030/X68000シリーズ

成熟するウィンドウ環境で

65,536色対応、動画ウィンドウ標準装備。

SX-WINDOW ver3.0 システムキット

NEW

CZ-294SS(5インチ版)
CZ-294SSC(3.5インチ版) 各19,800円(税別)

512×512ドットのエリア内で、自然描画に迫る美しい表現が可能な65,536色表示のグラフィックウィンドウを駆使できます。さらにグラフィックウィンドウ内でのアニメーション動画表示、各種グラフィックデータのコンバートも実現しました。高機能エディタ「日本語マルチフォントエディタ」を標準装備。アウトラインフォントの展開もフォントマネージャの効率化により、さらに高速化が図られています。その他、最大ズームサイズの設定や任意サイズのグラフィックを背景に設定できるなど、クリエイティブワークをサポートする数々の便利機能を装備しています。Human68k ver3.0システムディスクを付属しています。

※メインメモリ4MB以上必要です。※SX-WINDOW ver1.0/1.1/2.0をお持ちの方には有償バージョンアップを行います。

(日本語マルチフォントエディタの特長)

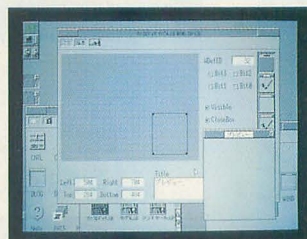
■自由なフォント設定:フォントタイプ、サイズ、スタイルを文字単位に指定可能。ルビも自由な大きさで付けられます。■ワープロ機能:禁則処理(追い出し、ぶら下がりも指定可能)、ワードラップ(半角文字)。■ユーザーカスタマイズ機能:キー割り当て、マクロ定義、メニュー定義(アイコンも定義可能)、外部コマンドなど。■イメージデータの貼り付け:パターンエディタなどで作成したビットイメージデータの貼り付けが可能。■シングルウィンドウモードの追加:複数のファイルをひとつのウィンドウで編集ができます。ファイルごとに編集環境の切り換えが可能。■その他:レイアウト機能の強化、矩形カット&コピー/矩形ペースト、マーク・ジャンプ機能。

待望のSX-WINDOW開発支援ツール。

SX-WINDOW 開発キット Workroom SX-68K

NEW

CZ-288LWD 開発中

SX-WINDOW用のソフト開発に必要な開発ツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解ができる33種のサンプルプログラム付き。また各マネージャ解説と関数リファレンスの詳細なマニュアルも装備しています。 ※メインメモリ4MB以上、SX-WINDOW ver2.0以上、C compiler PRO-68K ver2.1が必要です。

キット構成

■開発ツール

●SXデバッガ
SX-WINDOW上で複数のプログラムを同時にデバッグすることができるソースコードデバッガ。

●リソースエディタ
SX-WINDOW上のリソースをリソースタイプごとの編集ウィンドウでビジュアルに作成・編集が可能。

●リソースリンカ
Cコンパイラやアセンブラで作成したリソースデータファイル(オブジェクトファイル)をリンクしてリソースファイルを作成。

●サンプルメイク
サンプルプログラムのコンパイル作業をSX-WINDOW上から、XCver2.1のMAKE.Xを呼び出して、自動実行する簡易メイクユーティリティ。

■サンプルプログラム

●基礎編(23種)
各マネージャの基本的な機能のみを用いた基本動作の理解。

●応用編(4種)
基礎編での基本機能を応用した簡単なアプリケーションの作成。

●実用編(6種)
基礎/応用編での機能を駆使した、実用的なアプリケーションの作成。

■その他ファイル

●インクルードファイル
Cコンパイラとアセンブラ用の関数定義、データ定義ファイル。

●ライブラリファイル
Cコンパイラ用関数ライブラリ。

マニュアル
●ユーザーズマニュアル
●プログラマーズマニュアル
●SXライブラリマニュアル

新製品

X68030

32bit PERSONAL WORKSTATION

さらに高度な創造次元へ。

● SX-WINDOWを楽しく使うためのアクセサリ集

SX-WINDOW デスクアクセサリ集 NEW

CZ-290TWD 標準価格14,800円(税別)

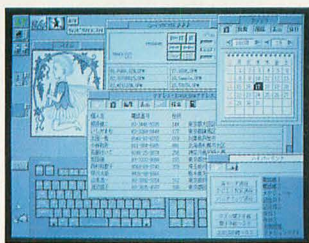
SX-WINDOWをさらに便利に、楽しく使うためのデスクアクセサリ集です。スクリーンセーバ、アドレス帳、電子手帳通信ツール、パズルなど12種類の豊富なアクセサリが収められています。

1キーノート2スクリーンセーバ3スクラップブック4ミュージックボックス5ハイパーリンク(電子手帳通信ツール)6アドレス7スケジューラ8ウィンドウアイコニファイ9ソフトウェアキーボード10パズル11ファイルサーチ(ファイル検索ツール)12フォントリンカ。 (2MB、ver3.0)

● SX-WINDOW対応になってさらにパワーアップ。

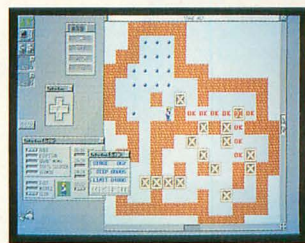
倉庫番リベンジ SX-68K ユーザー逆襲編 NEW

CZ-293AW(5インチ版)CZ-293AWC(3.5インチ版) 標準価格6,800円(税別)

10年にわたるユーザーの投稿など、新作306面が目白押し。まさに倉庫番の最強版がSX-WINDOW上で楽しめます。移動可能先が表示されるAI機能を搭載、またマウスをクリックするだけで簡単に問題を作成できるエディット機能や、キャラクタを替えてちょっと違った雰囲気でゲームが楽しめるキャラクタ変更機能も装備しています。半年で解けたらあなたは天才?です。 (2MB、ver1.1)

● マルチタスク機能をはじめ、通信環境がさらに充実。

Communication SX-68K

CZ-272CWD 標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションソフトを実行中でも簡単に通信が可能。また、ホスト局をクリックするだけの自動ログイン機能、初心者にも簡単なプログラム機能、最新モデム(20種類)もフルサポートしています。 (2MB、ver1.1)

● ウィンドウ対応グラフィックツール。

Easypaint SX-68K

CZ-263GWD 標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。 (2MB、ver1.1)

● 多彩なサウンドクリエイトを実現するFM音源サウンドエディタ。

SOUND SX-68K

CZ-275MWD 標準価格15,800円(税別)

他のミュージックソフトで演奏中の音色を、簡単に作成、変更ができるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。まさにミキサー感覚で音創りが楽しめるツールです。 (2MB、ver1.1)

●「SX-WINDOW開発キット」のサポートツール。

開発キット用ツール集 NEW

CZ-289TWD 開発中

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールの簡易リファレンスを簡単に検索するインサイドSX、イベントの発生を常時監視確認するイベントハンドラ、リアルタイムにメモリブロックの利用状況を表示するヒープビューアなど11種のツールが用意されています。 (2MB、ver2.0)

※(2MB、ver1.1)の表示は、メインメモリ2MB以上、SX-WINDOW ver1.1以上が必要であることを示します。

充実の PRO-68K シリーズ

● マルチフォント印字に対応。

Multiword ver2.0 NEW

CZ-225BSV

標準価格32,000円(税別)

Zeit社の書体倶楽部をサポート。同時に6書体のフォントが指定可能、レーザプリンタのフォントも複数使用できます。またキー操作やメニューの改良、均等割り付け、グラフィックのアイコン化なども可能。

※MultiwordおよびMultiword ver1.1をお持ちの方には有償バージョンアップを行います。

● ビジネスグラフチャート。

CHART PRO-68K

CZ-267BSD 標準価格38,000円(税別)

※以上のPROシリーズのソフトの動作にはメインメモリ2MB必要です。

※発売予定のソフトの画面写真は実物とは異なる場合があります。

●お問い合わせは…シャープ(株)電子機器事業本部AVCシステム事業推進室 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ シャープ株式会社

SHARP

入力から、編集、出力、複写まで、カラーの世界は、シャープです。

マインドに響く。

高品位クリエイティブワークツール for X68030/X68000シリーズ

600DPI※、1,677万色、高品位、高画質、高速読み取りを実現。

●基本解像度300DPI、当社独自手法により最高600DPIの高解像度読み取りを実現、微細な線や点も鮮明に再現。30～600DPIの範囲で最小0.01DPI単位の解像度指定と読み取り範囲の画素指定が可能●各色1画素あたり256階調(8ビット/画素)のデジタルデータ処理により、約1,677万色の美しい再現力●スキャナヘッド移動時間を短縮することにより、トータル読み取り時間を大幅に短縮(当社従来比約2/3)●画像の編集や加工などグラフィック環境を強力にサポートする専用ユーティリティソフトを装備●3タイプの透過原稿読み取りユニット(別売)で、A4から35mmまでのネガ/ポジフィルムなどの透過原稿に対応●SCSIインタフェース標準装備

※当社独自手法による擬似解像度

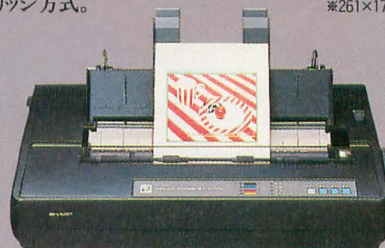
NEW

カラーイメージスキャナ
JX-325X
標準価格190,000円(税別)

OUTPUT

3種類の制御コマンドモードを搭載。質感鮮やか、高品位カラーイメージジェット。

シャープ独自のIOシリーズコマンド(Gモード)に加え、NM-9900モード(Nモード)、ESC/P24-J84C準拠モード(Pモード)をサポート。一般文書の作成から各種デザイン、建築用パースなどCAD分野に対応●発色性に優れた普通紙対応の新黒インキ採用。専用紙はもちろんオフィスでよく使われる普通紙にもカラー印字●プリントバッファメモリ(128KB)の内蔵で、ホストコンピュータの拘束時間を軽減●48ノズル(各色12ノズル)採用の高速印字。A4用紙1ページ※を約90秒でプリント(データ受信時間除く)●ビジネス用途に適したB4横用紙幅対応●OHPフィルム(専用)にも鮮明プリント●ノンインパクトならではの静粛印字●インキ補充は簡単、経済的なカートリッジ方式。

※261×174(mm)領域

カラーイメージジェット
IO-735X-B
標準価格248,000円(税別)

SHARPオリジナル
IO-735X-B 対応 アプリケーション

●SX-WINDOW対応ペイントツール
Easypaint SX-WINDOW
CZ-263GW 標準価格12,800円(税別)

●WYSIWYGを実現、ドローグラフィックソフト
CANVAS PRO-68K
CZ-249GS 標準価格29,800円(税別)

●オリジナリティを活かせるポップアップツール
NEW Printshop PRO-68K ver. 2.0
CZ-221HS 標準価格20,000円(税別)

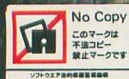
●マルチワープロ PRO-68K
Multiword ver. 2.0
CZ-225BSV 標準価格32,000円(税別)

CHART PRO-68K
CZ-267BSD 標準価格¥38,000(税別)

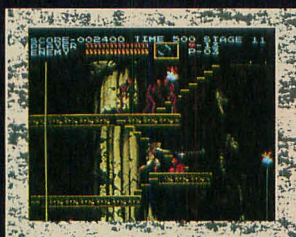
Press Conductor PRO-68K
CZ-266BSD 標準価格¥28,000(税別)

SX-Window ver. 3.0
CZ-294SS(C) 標準価格¥19,800(税別)

資料のご請求・お問い合わせはコンシューマーセンター
●東日本相談室…〒261 千葉市美浜区中瀬1丁目9番2号 ☎(043)297-1221(大代表)　●西日本相談室…〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表) シャープ株式会社

コナミ株式会社
本　社／〒105 東京都港区虎ノ門4-3-1　札幌営業所／〒060 札幌市中央区北一条西5-2-9
大阪支店／〒530 大阪市北区梅田1-11-4　福岡営業所／〒810 福岡市中央区天神2-8-30
●コナミの最新情報●
週間テレフォンサービス

ホラーアクションの名作「悪魔城ドラキュラ」が遂にX68000で登場。
かつてない完成度のグラフィックと臨場感あふれるサウンドで蘇る、血の戦慄。
ドラキュラシリーズ史上最高の闘いが今、はじまる。

百年の永い眠りより暗黒の野望を抱き
魔王ドラキュラがふたたびこの世に舞い戻った。
この危機を救おうと青年シモン・ベルモンドは一族に伝わる
不思議な力を秘めたムチを手に立ち上がる。
襲いかかる数々の罠とモンスターたち。はたして
魔王ドラキュラの野望を打ち砕くことができるのか。
ふたたび戦慄の死闘の中へ。

© 1986 1993 KONAMI ALL RIGHTS RESERVED.

7月23日発売予定 定価9,800円(税別)

MIDI POWER ver.3.0 X68000 COLLECTION 定価2,800円(税込) KICA7615

グラディウスIII(MIDIアレンジ)、悪魔城ドラキュラ(X68000オリジナルサントラMIDIアレンジ) 全42曲収録。
IKAchanのMIDI POWER講座、グラディウスIII タイトルBGM楽譜付。 制作・発売元/コナミ株式会社 販売元/キングレコード株式会社

関東 東京03(3436)2277 東北 秋田0188(24)7000 九州 福岡092(715)8200 大牟田0944(55)4444
関西 大阪06(456)4578 北陸 新潟025(229)1141 志布志0994(72)0606 鹿児島0994(44)3977
北海道 札幌011(241)4900 四国 松山0899(33)3399 (以上、全10局開設しております。)

●商品に関するお問い合わせは●
お客様ご相談室 コナミホットライン フリーダイヤル0120-086-573(ハロー コナミ)
営業時間:月曜日〜金曜日(祝日を除く)10:30〜17:00 電話番号は お間違えのないようにおかけください。

B L A C K

68 COLER JOY CONT TurboV

対応機種
シャープ　X68000、X68030シリーズ
某F社　FM-TOWNS、FM-TOWNSIIシリーズ
各社　MSXパソコン

7月19日より受付開始

1000本限定発売!!

送料、消費税込み3000円ジャスト！まずはお電話で整理番号を確認の上ご送金下さい。
受付電話番号東京03-3253-9183　受付時間午前10時～午後6時まで
今回特別サービスとしてお申し込み先着10名様に無料進呈致します。
たったの1000本だけの限定販売ですからお早めにお申し込み下さい。

- ■A,Bトリガー独立連射回路採用、ボリュームによって無段階に連射速度を調整できます。
- ■連射トリガーを押した瞬間に必ず弾が出るシンクロスタート連射回路搭載(PAT. PENDING)
- ■単射、連射独立トリガー方式採用。瞬時に使い分け出来ますので溜め撃ち、連射のどちらも出しやすい機能です。(実用新案1-80104)
- ■表面が平らで裏面が曲線を描いている持ちやすいフォルム

IVR SYSTEM MOUSE
Sygnas X
PRODUCED BY SPITAL

X68000待望のVRマウス「SYGNAS」ついに登場

68史上初のサードパーティマウス、純正を越える高機能、買い替えをするならシグナス

仕様
- ●分解能は400／300／200／100dpiの4段切替＋オート機能
- ●マウス側面のターボ・ボタンを押すと瞬時に800dpiの高分解能になります。
- ●オートは動かす速度に合わせて100→200→300→400dpiの分解能に滑らかに自動変速します。
- ●手ブレを50%カットするファジー・リバイズ機能搭載(軌跡補正回路)PAT.PENDING
- ●設定されている分解能が一目で分かるLEDインジケーター付、オートモード時は先端のLEDが点滅、ターボモード時は4個のLEDがフラッシングします。

型番IVR-68　標準価格**9,800**円(税別)

対応機種
シャープ　X68000、X68030シリーズ
姉妹品も同一スペックで同時発売
IVR-98　NEC98、EPSON PCシリーズ専用
IVR-FT　富士通FM-TOWNSシリーズ専用

惜しくも限定版が手に入らなかった方は申し訳ございませんがお近くのパソコンショップで市販タイプをお求めください。

市販タイプ　標準価格2,980円
あの有名なFM-TOWNSカラーのCT-5

SPITAL SANGYO CO.,LTD.
スピタル産業株式会社
〒101 東京都千代田区外神田1-16-1
TEL.03(3253)9183 FAX.03(3251)2946

安いのに親切 TSUKUMO

半期に一度の ツクモ決算大セール

ツクモグローバルカード

大人気！入会者募集中！ 18才以上なら学生さんもOK！

国内・外で活躍！使って便利、持ってて安心！ツクモグローバルカードはジャックス・VISAとの提携カードです。ツクモ各店でのお買物がらくらくできる上に、国内はもとより海外での分割ショッピングもOK！
20才以上の方にはキャッシングカードも発行致します。
お申し込みは☎03(3251)9898又は店頭にて！

シャープX68000の事なら何でも揃うツクモにおまかせ！
SHARP X68000コーナーは、パソコン本店II 3Fに。ゲームソフト関連は4FにOPEN！！

ついに登場

X68030

シリーズ最高峰。ユーザーの期待に応えて更にパワーアップしたX68030！
大好評発売中!!

- 新たに32ビットCPU(MC68EC030/25MHz)を搭載し、従来機の2.4～4.2倍以上のスピードアップを実現！
- 成熟するウィンドウ環境、使いやすさと高機能を追求し、動画機能、SX-WINDOW Ver.3.0搭載
- SX-WINDOWの操作環境を考え、4MBメモリ内蔵
- カラー液晶ディスプレイ接続可能

5インチFDDモデル	CZ-500C-B	標準価格¥398,000
5インチHDDモデル	CZ-510C-B	標準価格¥488,000
3.5インチFDDモデル	CZ-300C-B	標準価格¥388,000
3.5インチHDDモデル	CZ-310C-B	標準価格¥478,000

ツクモ特価販売中

大好評発売中！

おすすめの組み合わせ

CZ-500C-B……¥398,000
240MBハードディスク……サービス
ツクモ決算特価¥388,000
★★★★超速い★★★★

X68030用8MB増設RAMボード発売!!
●これ一枚でいっきに12MBフル実装●
SH-5BE4-8M…ツクモ決算特価¥46,800

X68000、X68030用ドライブTSシリーズ大好評発売中!!

目のつけどころがツクモでしょう。

●X68000 & X68030シリーズ対応3.5インチフロッピーディスクドライブ

TS-3XRシリーズ
〈仕様〉
- 3.5インチ2DD/2HD/2HCフォーマット対応
- ユーティリティソフト付属（デバイスドライバー／フォーマッター）
- 標準サイズケーブル付

TS-3XR1(1ドライブ) 定価¥44,800 ツクモ決算特価¥35,800
TS-3XR2(2ドライブ) 定価¥57,800 ツクモ決算特価¥46,800

●Compact XVI/X68030シリーズでお使いの方は、別売ケーブル(TS-XR5CA特価¥6,800)が必要です。

●X68000Compact & 68030シリーズ対応フロッピーディスクドライブ

TS-5XRシリーズ
〈仕様〉
- 5インチ2HD/2DDフォーマット対応
- ドライブ番号切り換スイッチ付
- Compact XVI/X68030用ケーブル付

TS-5XR1(1ドライブ) 定価¥53,800 ツクモ決算特価¥42,800
TS-5XR2(2ドライブ) 定価¥72,800 ツクモ決算特価¥57,800

X68000の5インチモデルをお持ちの方へ!! おすすめ「X68030セット」

「ケーブル1本でX68000が5インチドライブとして使えます！」という訳で、X68030購入をお考えの方ならばこの組み合わせ

CZ-300CB……¥388,000
TS-XFDCA……¥9,800
合計定価¥397,800 ツクモ決算特価¥318,000

コンピュータアート

スーパーグラフィクツールセット

その1. 慣れてしまうとマウスがいらない
NS Calcomp製 Drawing Pad(タブレットセット)……¥76,500
サンワード Matier(マチェール)……¥39,800
合計定価¥116,300 ツクモ決算特価¥95,000

その2. ハイクオリティなのにこんなに安い
ヒューレットパッカード HP Desk Jet 505J(インクジェット)……¥99,800
ヒューレットパッカード カラーキット……¥12,000
アーベル プリンタケーブル……¥4,800
サンワード Matier(マチェール)……¥39,800
合計定価¥156,400 ツクモ決算特価¥112,000

～～～ 旧製品も大特価 ～～～

台数限定

X68000XVI CZ-634C-TN ツクモ決算特価¥148,000
X68000Compact XVI CZ-674C-H ツクモ決算特価¥138,000

おすすめSCSIタイプハードディスク

120MBハードディスク……ツクモ決算特価¥50,000
170MBハードディスク……ツクモ決算特価¥63,000
240MBハードディスク……ツクモ決算特価¥75,000

X68000シリーズ用RAMボード

1MB増設RAMボード・ツクモ決算特価¥11,000 (CZ-600C専用)
1MB増設RAMボード・ツクモ決算特価¥11,000 (ACE/PRO/PRO2シリーズ用)
2MB増設RAMボード・ツクモ決算特価¥23,000 (拡張スロット専用)
4MB増設RAMボード・ツクモ決算特価¥39,000 (拡張スロット専用)

大容量記憶装置

MOが今一番トレンディ
★Logitec 3.5インチ光磁気ディスクユニットセット★
LMO-FMX330……¥178,000
SCSIケーブル……サービス
ツクモ決算特価¥148,000
※MOメディア レンズクリーナー、フィルター付属。
※Human 68K Ver.3.0以上が必要です。

パソコン通信時代は14400ボーへ

★モデム
AIWA PV-AF144V5 定価¥64,800 ツクモ決算特価¥49,800
★通信ソフト
- たーみのる2……ツクモ決算特価¥13,000
- Communication SX-68K・ツクモ決算特価¥16,800

MIDIコンピュータミュージック特選セット

特選Aセット
- SC-55MKII…¥69,000
- SX-68MII……¥19,800
- Mu-1 Super…¥39,800

合計定価¥128,600 ツクモ決算特価¥99,000

特選Bセット
- CM-500……¥115,000
- SX-68MII……¥19,800
- Mu-1 Super…¥39,800

合計定価¥174,600 ツクモ決算特価¥140,000

超低金利！冬・夏ボーナス二回払受付中！詳しくは各店までお問い合わせ下さい。

通信販売のご注文は下記フリーダイヤルへ。
全国どこからでも通話料無料
受・注・専・用 フリーダイヤル 0120-377-999
通販センター 03-3251-9911 商品についてのお問い合わせは各店又は通販へ。

クレジット払い	各種リース払い	現金書留払い
月々¥3,000以上の均等払いも頭金なし、夏・冬ボーナス2回払いも受付中！	くわしくは各店にお問い合わせ下さい。ケースに合わせてご相談にのります！	〒101-91 東京都千代田区神田郵便局私書箱135号 ツクモ通販センター Oh!X係
カード払い(¥5,000以上)	**全国代金引き換え配達**	**銀行振込払い**
通信販売での御利用カード、ツクモグローバルカード、VIPカード、セントラル、ジャックス＊御本人様より電話で通販部へお申し込み下さい。	お申し込みは☎03-3251-9911へお電話1本！配達日の指定もできます。	事前に☎でお届け先をご連絡下さい。三和銀行 秋葉原支店(普)1009939 ツクモデンキ

ツクモ全店7月は無休で営業いたします。

秋葉原各店 営平日AM10:45～PM7:30 日・祝AM10:15～PM7:00

ツクモパソコン本店II 3F
☎03-3253-1899(直通)(担当／荒井)
ツクモパソコン本店代表☎03-3253-4199 休毎週木曜日
ツクモニューセンター店 ☎03-3251-0987(担当／沢栄) 休毎週木曜日
(下取り交換、中古販売も行っております。)
※定休日が祝日と重なる場合は営業致します。
※8月11、12日は臨時休業させていただきます。

パソコン本店 荒井

名古屋各店 8 10、11は営業いたします。
名古屋1号店 ☎052-263-1655
営AM10:00～PM7:00 休毎週火曜日(8 16～18休業)
名古屋2号店 ☎052-251-3399(担当／松原)
営AM10:00～PM7:00 休毎週水曜日(8 18～20休業)

札幌各店 8 11、12日は臨時休業させていただきます。
ツクモ札幌店 ☎011-241-2299(担当／田口)
営AM10:30～PM7:30 休毎週木曜日
DEPOツクモ2番街店 ☎011-242-3199(担当／鈴木)
営平日AM10:40～PM7:30
日・祝日AM10:10～PM7:00 休毎週木曜日

ツクモは「スーパーX PRO SHOP」です。
PRO STAFF ツクモ
九十九電機株
〒101-91 東京都千代田区神田郵便局私書箱135号
★商品のご注文は在庫確認の上お願いします ★表示価格には消費税は含まれておりません

パソコンショップ満開　本格的開店！人気ゲームソフト大特価！
さらにあやしい企画やウハウハなハードウェアもぞくぞく開発中なのだ

X68030&コプロセッサ

- ・CZ-500C(5"FDモデル本体) ¥??8,000（税別）
- ・CZ-510C(5"FD 80MBHDモデル本体) ¥3?6,000（税別）
- ・CZ-300C(3.5"FDモデル本体) ¥??1,000（税別）
- ・CZ-310C(3.5"FD 80MBHDモデル本体) ¥3?8,500（税別）
- ・MC68882FN25A ¥16,000（税込）
  (CZ-5MP1同等品、取付図解付)

REDZONE&XVI&Compact

- ・REDZONE(CZ-674C改) ¥160,000（税別）
  (10、16、24MHzの3モード)
- ・CZ-634C(XVI) ¥168,000（税別）
- ・CZ-674C ¥140,000（税別）

周辺機器&RAM

- ・30用4MBRAMボード(CZ-5BE4) ¥42,000（税別）
- ・30用4MBRAMモジュール(CZ-5ME4) ¥38,000（税別）

- ・14型ディスプレイ(CZ-608D) ¥68,000（税別）
- ・14型ディスプレイTV(CZ-607D) ¥68,000（税別）
- ・15型ディスプレイTV(CZ-614D-TN.BK) ¥101,000（税別）
- ・外付5.25FDドライブ(CZ-6FD5) ¥49,800（税別）
- ・満開式軟盤駆動装置壱號(MK-FD1) ¥39,800（税別）

各機種用キーボードも扱っております。お問い合せ下さい。
★のマークのついたソフトは十分な在庫が確認できませんので代金引換のみの取扱となります。
●印のないソフトは5インチのみの取扱となります。
マークのないソフトは在庫が十分ございますので、ご安心してお申込になれます。

ソフトウェア

- ・SX-WINDOW ver3.0
  (CZ-294SS 5インチ版、CZ-294SSC 3.5インチ版) ¥17,000
- ・SX-WINDOW デスクアクセサリ集
  (CZ-290TWD 3.5、5インチ同梱) ¥12,500
- ・Communication SX-68K
  (CZ-272CWD 3.5、5インチ同梱) ¥17,000
- ・Easypaint SX-68K
  (CZ-263GWD 3.5、5インチ同梱) ¥10,900
- ・SOUND SX-68K
  (CZ-275MWD 3.5、5インチ同梱) ¥13,400

以上のソフトは消費税・送料込みの価格です。

ゲームソフト

ゲームソフトは税別、送料サービスです。

- ・倉庫番リベンジSX-68K
  (CZ-293AW 5インチ版) ¥5,400
  (CZ-293AWC 3.5インチ版) ¥5,400
- ・ボスコニアン ¥5,800
- ・ファンタジーゾーン ¥6,700
- ・モトス ¥6,700
- ・バブルボブル ¥6,100
- ・ギャラガ'88 ¥7,000
- ・エイリアン・シンドローム ¥5,000
- ・キャメルトライ ¥7,500
- ・イース ¥8,100
- ・苦胃頭捕物帳 ¥4,500
- ・テラクレスタ・ムーンクレスタ ¥6,800
- ・チェルノブ ¥4,200
- ・スターフォース ¥4,200
- ・リブルラブル ¥6,800

- ★ポピュラス ¥7,800
- ★ポピュラス2 ¥10,000
- ★プロミストランド ¥4,000
- ★シムシティー ¥7,800
- ★シムシティー テレインエディター ¥4,000
- ★パワーモンガー ¥10,000
- ★プロサッカー68 ¥7,800
- ★レミングス ¥6,200

- ★●天下統一 ¥7,800
- ★●ブルトン・レイ ¥7,000
- ★●遊撃王IIエアーコンバット ¥7,000
- ★●ブルトン・レイシナリオエディタ ¥4,000
- ★●ブリッツリーク ¥7,800
- ★マスター・オブ・モンスターズII ¥7,000

- ★A-JAX ¥7,000
- ★クオース ¥5,400
- ★生中継68 ¥7,800
- ★出たな!! ツインビー ¥7,800
- ・悪魔城ドラキュラ(7月23日発売予定) ¥7,800

- ★ジェノサイド ¥7,000
- ★ラグーン ¥7,000
- ★ファランクス ¥7,000
- ★ジェノサイド2 ¥7,000
- ・ストライダー飛竜 ¥7,800

- ★飛翔鮫 ¥7,000
- ★究極タイガー ¥7,000

通信販売の方法

★お支払いと商品のお届け方法
●現金書留、郵便振替のいずれかの場合、ご入金確認の後、在庫があれば1週間以内に発送いたします。
●代金引換え(着払い)にてもお受けいたします。
●商品到着後1週間以内の初期不良は新品交換いたします。
●すべて現金一括払いのみの取扱いとさせていただきます。
●返品は到着後5日以内に未開封で返送料はお客様負担でお願いいたします。なお、その際は事前に電話連絡をして下さい。
★現金書留または、郵便振替の場合下記の宛先へ代金をお送りください。
現金書留の場合：〒171 東京都豊島区長崎1-28-23Muse西池袋2F パソコンショップ満開
郵便振替の場合：東京 5-568201 パソコンショップ満開
●ご注文の際は、郵便番号・住所・氏名・電話番号を必ずご記入下さい。
★お問い合わせ先 TEL(03)3554-7441〔月～金 午前11時～午後6時〕

パソコンショップ満開

TEL 03-3554-7441
郵便振替 東京 5-568201
口座名 パソコンショップ満開
※住所・FAXは満開製作所と同じです。

(株)満開製作所

〒171 東京都豊島区長崎1-28-23
Muse西池袋2F
TEL 03-3554-9282
FAX 03-3554-3856

THE USER'S WORKS

●STRIP FIGHTER Ⅱ'（ストにゃんダッシュ）

神裸羞殺拳の正統奥義伝承者の座を賭け，9人のプレイヤーからひとりを選んで世界中を対戦していく。タイトルから想像されるようなストⅡタイプのHな格闘ゲーム……ではなくて格闘型のカードゲームだ。

●

ゲームシステムは，はっきりいえば某バトルスキンパニックと同様なものと思っていい。基本的にはカードを順番に出しあって相手を攻撃する。なかには防御値を上げたり，体力を回復したりするカードもある。使ったカードは随時補充されていく……というと配牌の運だけで決まりそうなつまらないゲームにも思えるが，このゲームではカードの出し方で格闘するという感覚をうまく再現している。

まず，通常の攻撃カードにはお約束どおりパンチとキックが弱中強の3種類ずつある。となれば，大技のあとにはスキができやすい，というのも常識であろう。

同時に1枚または2枚のカードを場に出せる。このカードの組み合わせによって生まれるのが「必殺技」だ。必殺技の出し方には大きく分けて，前に出したカードとのシーケンスによるものと2枚のカードの組み合わせによるものがある。

なんのカードも出さずに自分のターンを終了すると「タメ」となり，カードを入れ替えることで「待ち」が使える。場合によっては相手の必殺技（シークェンス系）を封印することも可能だ。

必殺技とは？

やはり気功である。達人ともなれば「気」を使いこなすことで，指1本で相手の肉体を粉々にしたり，息を数千度の炎にして吐き出したり，上昇中無敵の必殺拳を放つことができるのは周知のことであろう。

このゲームでも同様な設定が行われている。自然の力（外気）を吸収し，感情（内気）を高揚させることによって増幅する。この内気として猫娘の羞恥心を応用したのが神裸羞殺拳である。恥ずかしい思いをすればするほど強くなるのだが，なにぶん，うら若い猫娘なので度を越えて恥ずかしくなると戦えなくなる。

ということで，必殺技は大きく「痛い技」と「恥ずかしい技」に分類される。

マニュアルとともに必殺技一覧も付属しているが，最初はそれを見ないでやってみることをおすすめする。試行錯誤するなり，相手から受けた技を体で覚えていくのもいいだろう。技を探す楽しみも味わえる。

勝負の決め手になるのはあくまで体力なのだが，相手の羞恥心を上げきってしまえばあとは殴るだけ。いきおい必殺技中心のゲーム構成になってくる。

さて，必殺技の存在はこのゲームの性格を微妙にしているように思われる。攻撃力などは「脱衣」で強化されるのだが，必殺技を使っていればむしろ脱衣しないほうが有利に試合を展開でき，Hゲームである必然性は薄くなっている。格闘型カードゲームで脱衣「も」できるといったほうが正確であろう。そういったこともあってか，このゲームではクリア後のグラフィックを表示しないモードもついている。それもひとつの道であろう。

通常のゲームと対戦モードのどちらかを選び，使用するキャラクターを決める。なぜか日本代表って変な技ばかり使う。

とはいえ，使用するキャラクターにもよるが，必殺技で押していけば相手が脱衣する前に片をつけるのは簡単だ。いかに長引かせて……というのが，まあ，このゲームの本道であろう。

なお，このゲームには人間対人間の対戦モードも用意されている。シークェンス系の技は封印されることもあるので，同時系の必殺技を持つキャラクターが有利だ。対戦ではドイツ猫のピュティアが最強ではないかと思われる。ほとんどの配牌から恥ずかしい必殺技を繰り出せるのだ。

入手方法

注意点はひとつだけ。このゲームは要2Mバイトである。システムはHuman68k ver. 3.0なのでX68030でもそのまま起動できる。ハードディスクへのインストールも可能だ。

入手希望者は下記住所まで郵便で連絡してほしい。書面に希望するゲーム名などを明記し，1,300円分の無記名の定額小為替（郵便局で買う）と自分の住所を書いたタックシール（ディスクラベルでも可）を同封することが望ましい。なお，5インチか3.5インチかの指定を忘れないように。

〒182　東京都調布市富士見町4-9-46
エスペランサ西調布106　南国猫桜

上左が通常モード，上右が対戦モードの画面。相手を倒せばお約束のグラフィックが見られる。

必殺技を駆使して並み居る敵を倒すと，感動（？）のエンディングが待っている。

SOFTWARE INFORMATION

期待の新作「コットン」が正式にリリースされた。発売はちょっと先だけどファンにとっては嬉しいニュースだ。また，これから秋に向けての新作ソフトがいろいろあるので，しっかりチェックしておこう。

コットン

かねてから移植希望の多かった「コットン」が，X68000に移植されることが決定した。

主人公は食いしん坊でお調子者の魔法使いコットン。そのコットンが，闇に閉ざされてしまった妖精界に光をもたらす救世主となるべく……な～んてストーリーがトントン拍子に進むわけもなく，当のコットンはまるっきり無関心。だが，妖精シルクから魔物を退治すると大好物の“WILLOW(ういろう)”が手に入ると聞いたとたん，空飛ぶホウキを手に目の色変えていざ出陣！　色気より食い気，地位や名誉よりもビックな“WILLOW”のためにコットンは戦うのだ。

とこんな感じでストーリーが展開していく。

上位は変化なしだが，来月は……

		(前回順位)
1．	悪魔城ドラキュラ	1
2．	リブルラブル	2
3．	SX-WINDOW開発キット	3
4．	餓狼伝説	4
5．	ストリートファイターII	5
6．	コットン	—
7．	EG Word	7
8．	クレイジークライマー1＋2	—
9．	ロボットコンストラクションR.C.	—
10．	マージャンクエスト	9

1位はやはり「悪魔城ドラキュラ」。そのあと5位までは，まったく前回と変わりありません。このうち，「悪魔城ドラキュラ」「リブルラブル」「餓狼伝説」の3本は，この号と前後して発売されます。

しかし，「SX-WINDOW開発キット」ははっきりとした発売日は決定しておらず，「ストリートファイターII」などは発売すら決まっていないのですから，次回もトップ10入りを果たすのではないでしょうか。

6位は新登場の「コットン」。これは上で紹介していますが，アーケードゲームからの移植です。発売元はEAビクター。

内容は，かわいい魔女がホウキに乗って，飛び回り撃ちまくるという横スクロールシューティングゲームで，一部に熱狂的なファンを生みました。X68000ユーザーの間でも，以前から移植希望の声も多く，それに応えたというところでしょうか。

8位の「クレイジークライマー1＋2」は，“ビデオゲームアンソロジー”シリーズの次回作です。「リブルラブル」に例のパッドがつくと聞いたときに，「もしや」と思った人も多かったことでしょう。

植木鉢や鉄骨をよけながらビルをよじ登るという，これも「リブルラブル」同様，ジャンル分けの難しいゲームです。

残る新登場ソフトは，9位の「ロボットコンストラクションR.C.」。ユニークな題材ながら見かけが地味なだけに，発売前の人気がどうなるかは謎でしたが，そこそこの反響を得ているようです。推薦理由を読んでいると，やはり，ロボットを組み立ててプログラミングができる点が注目されているようです。

このミーハーチックなノリのよさ，テンポのいいゲーム展開にゲームセンターでハマった人も多いことだろう。また，ちょっぴり恥ずかしかったビジュアルシーンも，自宅でなら思いっきり悶絶できるというものだ。

しかし，外見はミーハーでも中身は連射バリバリの硬派な横スクロールシューティングゲーム。なめてかかると，見た目はかわいくてオチャメなキャラクターたちに，手痛いシッペ返しをくらうことになるだろう。

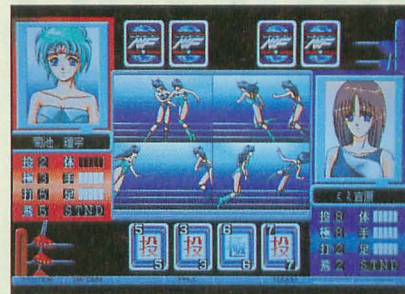
攻撃方法は，ショット，ボム，そして封印石を手に入れることで使用できる4種類の強力な魔法，さらに，敵に捕われている妖精を助け出すことでオプションも装備できる。ショットやボムは，敵を倒すと出現するクリスタルやアイテムでパワーアップし，魔法のビジュアルの派手さも忠実に再現。これは，現段階のサンプルでもなかなかの完成度といえる。まだ，ビジュアル関係はついていないものの，“テッテケテー”などのサンプリング音も健在だ。

発売は9月24日ということなので，この期待の1作を楽しみに待っていよう。また，新しいサンプルが届きしだい，追ってレポートする予定だ。

X68000用　5″2HD版2枚組　9,800円(税別)
EAビクター　☎03(5410)3111

レッスルエンジェルス

「女子プロレスリング」を題材にしたカードバトルゲーム，それが「レッスルエンジェルス」だ。試合は，飛び技，関節技，特殊カードなど5種類のカードを同時に出し合い，勝負を決めていく。シングル対戦，タッグマッチ，世界をまたにかけるストーリーが用意されているが，まずは，シングル対戦で技の出し方を研究するといいだろう。また，お決まりの脱衣シーンも用意されている。

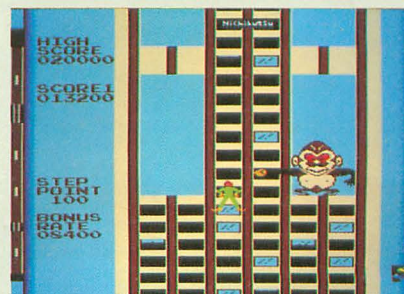
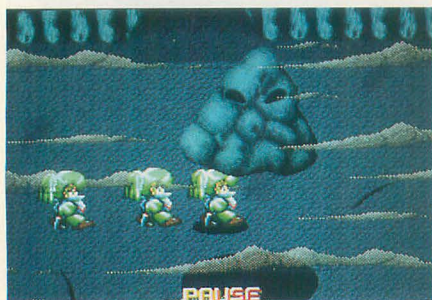
X68000用　3.5/5″2HD版4枚組　4,900円(税込)
ブラザー工業(TAKERU)　☎052(824)2493

クレイジークライマー1+2

“ビデオゲームアンソロジー”シリーズも順調に発売され，第5弾も順調に開発が進んでいるようだ。

その第5弾のターゲットとなったソフトは，タイトルを見れば一目瞭然,「クレイジークライマー」とその続編「クレイジークライマー2」がワンセットになったもの。ここにまた，ニチブツの名作ゲームが移植されることになった。古くからのゲーマーなら，あの“あ〜れ〜”に代表される効果音，BGMが耳にこびりついていると思う。

さて，プレイヤーの目的は，右手と左手に対応したスティックをコキコキ動かして，高層ビルディングを登っていくことである。理由などない，そこにビルがあるから登るのだ。それがクレイジークライマーの宿命なのだから。もちろん，プレイヤーを妨害する黒い組織も存在する。閉じる窓，植木鉢を落とす住人，突然降ってくる看板，挙げ句の果てにキングコングまでプレイヤーをビルから蹴落そうと，必死に妨害する。次々と迫りくる妨害工作をかわし，安全な足場（この場合は手掛かりか？）を見つけながら，プレイヤーはひたすらビルを登り続け最上階を目指す。そして，ビルの頂点に達したプレイヤーは，また，新たなるビルの頂点を目指すのだ。

操作に関しては，ジョイスティック2本でプレイすることはもちろん，「リブルラブル」に付属していたジョイパッドの対応もしているので，操作感覚も忠実に再現されるだろう。

X68000用　5″2HD版　価格未定
電波新聞社　☎03(3445)6111

TREND ANALYSIS

1993年7月号のハガキ集計ベスト10 最近買って気に入ったソフトは?

POINT	タイトル	発売元	発売日
82	エトワールプリンセス	エグザクト	'93/3/26
77	SX-WINDOW ver.3.0	シャープ	'93/3/30
35	スターフォース	電波新聞社	'93/3/26
33	オーバーテイク	ズーム	'92/11/20
13	テラクレスタ/ムーンクレスタ	電波新聞社	'93/1/29
12	MATIER	サンワード	'92/10/9
11	倉庫番リベンジ SX-68K	シャープ	'93/5/25
10	同級生	エルフ	'93/2/10
9	チェルノブ	電波新聞社	'93/1/29
8	Winning Post	光栄	'93/5/30

(無作為抽出した1000通のハガキを集計)

4位までは前回とまったく変わらない。とりあえず，これらは現在の定番ソフトといってしまってもいいのだろう。

1位の「エトワールプリンセス」のポイント数はほとんど変化が見られないが，2位の「SX-WINDOW ver.3.0」は伸びを見せ，両者の差はほとんどなくなっている。システムというものがものだけに，これからも大幅な落ち込みはないだろうから，ほかのソフトの動きによっては1位になることもあるかもしれない。

今回のSX-WINDOWのバージョンアップは，65535色や動画のサポートなど，派手めの話題も多く，発売される予定のソフトも「SX-WINDOW開発キット」や「Easy draw SX-68K」「EG WORD」など，期待の大きいものばかり。人気を保っているのは，本体だけではなく周辺の盛り上がりも関係していると思われる。

5位にはポイントは変わらないが，順位は上がってしまった「テラクレスタ/ムーンクレスタ」がいる。が，6位の「MATIER」とともに，ほとんど変化なしといっていいだろう。

7位の「倉庫番リベンジ SX-68K」は，SX-WINDOWで動くパズルゲーム。古くからのゲームファンなら誰でも知っているタイトルであろう。発売直後にしては，若干人気が薄い。内容的にはルールなどはそのままだが，システムはウィンドウシステムの特徴をうまく生かしている。実際のゲーム画面，マウスでパッドの代わりをさせる移動ボタン，キャラクター設定パネルなどがマルチウィンドウ化され，オマケに小物ゲーム，ツール類（実用的なものは少ないが）もたくさんついている。

8位と9位には，「同級生」と「チェルノブ」が返り咲いている。

10位には「Winning Post」が新登場。最初にアナウンスされた発売日からは何日か遅れてしまったが，約1週間後に無事発売された。

競馬シミュレーションというジャンルのゲームはX68000では初めてであり，最近はパソコンゲームでもコンシューマ向けでも人気の高い分野である。内容は単なるギャンブルゲームではなく，馬主を演じるという，どちらかというと経営シミュレーションに近いもの。いかにも光栄らしい仕上がりといっていいだろう。

今回は書籍関係が4.4%，特になしの人は45.9%という，ともに前回よりも少なめの割合になっている。

さて，恒例の次回予想をやっておこう。この号が発売された直後の7月23日に2つのゲームが発売される。「悪魔城ドラキュラ」と「餓狼伝説」である。前人気としては「悪魔城ドラキュラ」のほうが上回っているようだが，「餓狼伝説」のほうは発売日の延期の連続に待ちくたびれたという感があるので，実際に発売されたときにどうなるかはわからない。結果が楽しみだ。

ウワサのソフトウェア（海外編）

Scenery Animator 4.0

この風景の画像を自動生成するソフトは本誌にも過去に何回か登場して，お馴染みになっていることだろう。今回は作成した地形の中に，家や飛行機，熱気球など，好きな3Dオブジェクトを配置できるようになった。「Scenery Animator」は新たな局面に入ったといえよう。ポリゴンの物体とフラクタル地形の混在が可能になり，新たな可能性を開いたのだ。

物体はサンプルでいくつか提供されているので，とりあえずはそれでも遊べる。物体をモデリングする機能は備わっていないが，フォーマットが「VIDEO SCAPE 3D」という標準的な3Dソフトの形状ファイルなので，ほかの3Dソフトで物体を作ったオブジェクトを持ってくることができる。このあたりは，3Dの形状データのフォーマットを共有することについて，高い意識ができていることの強みであろう。以前に紹介した「Caligari」(操作性と表現力の両方を実現させた驚くべきモデラを持つ)や「PIXEL 3D」(2Dの画像に厚みをつけて，3Dの形状データに変換してしまう)，数多く市販されている3Dオブジェクト集など，簡単に3Dオブジェクトを作る(または入手する)環境がAMIGAには揃っている。

もともと，アニメーション生成機能があるので，3Dオブジェクトは置けるだけでなく，当然動かすことができる。方向とスピードを指定してまっすぐ飛ばすといった単純な運動だけだが，ピッチ角やバンク角もつけられ，たとえば「グランドキャニオンの谷間を飛行するステルス戦闘機」のようなものなら楽勝でできてしまう。

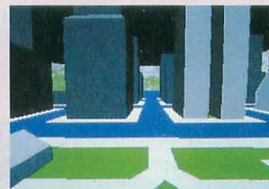
フライスルーのパス，つまり視点の移動経路のデザインは簡単で，しかも，かなり複雑な動きをさせることが可能。地図の上に適当な点をぽんぽんと打っていけば，その間を直線，または滑らかな曲線でつないでくれる。視点の位置はもとより視線の方向や視野の広さ，それにバンク角やピッチ角は自由に制御でき，さらに，計算機に任せて滑らかに変化させることも可能。自動計算はパスの曲線の傾きを見ながら行っているようで，自然な飛行感覚を簡単に表現できる。パスをループさせる機能は，エンドレスのアニメーション作成にうってつけだ。

作った動きはリアルタイムにプレビューできる。簡易表示ではあるものの，秒間数コマとなかなかの速度。モーションデザインの環境としても，なかなかいいセンいっている。プレビューの時点で，地面と視点の衝突判定もして教えてくれるので，計算してみると地面にめり込んでいたという間抜けなことは起きない。

全体の操作はあっけないほど簡単。画面上のボタンを好奇心に任せてつついているうちに，マニュアルなしでも機能が使いこなせるようになっていく。各機能に対しては，プログラマの立場から見ても感心させられる。内部でどうやっているのかわからないが，すごいことをさりげなくやっているのだ。広大な空間の中に点在する木々と3Dオブジェクトがきちんと共存し，しかも空間の中のどこをどの方向から見ても破綻しない。それでいて，アニメーションを実用的な時間内で作成することができる(最低限68030は装備したいが)。パソコンでこんなことができるとは，まったく驚異的である

ところで，競合製品である「VISTA PRO」がver.3.0になっているせいか，こちらは3.0をすっとばして4.0になってしまった。しかし，その数字に違わず，実際に感心するほどの発展を遂げているところがすごい。機能向上と同時に，使っていて楽しい気持ちを大事にしていることが感じられるのもうれしい。　(A.T.)

発売元　Natural Graphics

THE SOFTOUCH

ムチとロウソクと俺

Yokouchi Takeshi
横内 威至

いよいよ，発売まで秒読み段階に入った「悪魔城ドラキュラ」。全ステージにわたって繰り広げられる数々の演出にプレイヤーは釘づけになるだろう。ここにまた，新しい伝説が生まれたといっても過言ではない。

発売までついにあとわずか。これを手に入れずにもう生きてはいけない。人としてやらねばならない，守らねばならないことのひとつとして，ドラキュラシリーズを墓まで持っていくことが挙げられる。

買え！　ショップで最後の1本が取り合いになったら，アッパー昇竜を食らわせてでも自分のものとしろ。このときばかりはハメ殺しを許す。絶対に買い逃すな！

ドラキュラ復活

もともとは1986年のファミコン版がオリジナル。あらゆる経験を経て，ついに究極のドラキュラへといたる。そこで，久しぶりにファミコン版をプレイしたのだが，このX68000版とはまったく違っていた。当たり前か。このX68000版を見る前，俺ははっきりいって不安を感じていた。古くからのドラキュラフェチならおそらく不安を感じていたと思う。シリーズは回を重ねるごとに進化し続けたのであり，深く愛した人間はもうそこそこのドラッグじゃイケないと感じる。しかし，名作とはいえ初版の移植では，もしかしたら……。

だが完全に俺の思いすごしであった。やはりコナミがやるだけあって，ハンパなものではなかった。完全オリジナルである。進藤さんのおっしゃるとおり，画面効果，演出は現在をも超越する神の領域に達し，ゲーム内容，構成ともに練り直された，泣く子を感動の涙でさらに加速して泣かせる鬼のようなゲームになっている。

止まらずにさっさと進め

ハードコアへの誘い

操作が変わった。いや，むしろ拡張して加速したって感じだ。心あるドラキュラマニアなら，シリーズそれぞれ微妙な操作感覚の違いがあることを知っている。そのため，これから始めようという輩はもちろん，フェチのあなたも苦労するであろう。

だが安心しろ。慣れれば確実に動け，この新しい快感から逃れられなくなるに違いない。さあ，しっかりレザー，さらに加速してラバー，ボンデージで身を引き締めてバーチャルリアリティ・ムチ制御機器のジョイスティックを握ろう。

まず，操作の基本のジャンプであるが，いままでと異なり空中で自由に動ける。だから垂直に跳んでから前進することも可能だ。だが，ありがちなことに一度動くと止まれない。さらにジャンプしての下打ちもあり，空中でのレバー制御にかなり気を遣うのだ。俺もかなりのムチ使いであったつもりだが，慣れるのに結構てこずった。と同時に以前のシリーズがやりにくくもなった。やはり新しい感覚である。おっと，ひとつ注意してほしいのは頭のひっかかりだ。ジャンプして頭が壁なんかにぶつかると，それ以上跳べないし動けない。いけそうでいけないことが結構あるから注意。

当然ムチも微妙に違う。まずスピードに慣れること。ボタンをヒットしてから，どのくらいで振り切るかをしっかり体に叩き込むこと。これがわからないと絶対に遊べない。同時にムチの判定もよく叩き込む必要がある（X68000版は結構広い）。ファミコン版ではやや不安があったこの判定もまったく気にする必要がなくなった。かなり引き寄せてからでも間に合うのだ。斜め打ち，下打ちも予想以上に判定がある。かなり使う必要があるので，これは自由にできるようにしっかりマスターしよう。

さて，ひととおり操作がわかったら実践脳殺テクニックを身につける必要がある。まず，前進をしつつ攻撃できる“斜め打ち”を駆使できるようになろう。自分が止まっていては，ただ敵を溜めてしまい流れが悪くなる。といって，すべての敵にいちいち対処していてもいけない。歩きながらムチを振ると自分は止まってしまい，結局次々と敵が現れてしまうのだ。これは，慣れて

X68000用　5"2HD版2枚組　9,800円（税別）
コナミ　☎03(3432)5526

サルにしか見えない臭そうな男たちで稼げる

ブロックⅠのボス。斧で楽勝

くると結構かったるくなってくる。となると，必然的に斜め打ちを多用することになる。流れを止めず，美しく殺しながら進め。実際，自由に動かせて臨機応変に対応できることが，最も重要なことだからだ。

もうひとつ有効だといえるのが“流し打ち”とでもいうものだろうか。ジャンプして普通に打ってもだいたい最高点の高さに水平に打てるだけ。その高さの所に敵がいる瞬間でなければ意味がない。だが，落下中にムチを打てばムチが伸びたまましばらく落下する。つまり，その範囲にいた敵のすべてをしばけるのである。よく動く狙いにくい敵，たとえばコウモリなどには最適の方法である。このコウモリを止まって狙うより，はるかにやりやすく安全だ。後半では当たり前のしばき方だからタイミング，間合を覚えるように。サインカーブを描くメデューサの首を確実にしばけるようになっていれば，あなたはもう女王級。

10点。プログラマの遊びか？

目玉がエグイ上ルート

どんどん短くなるし火も飛んでくる

傾いたらジャンプして攻撃だ

美醜の館で一夜を

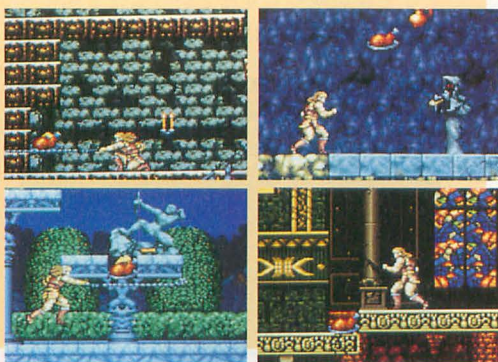
どうやらこのゲームは難しいらしい。俺は，ノーミスクリアするほどにまで成長したが，初心者には厳しい。しかし，悪魔城に秘められる数々の演出は，あまりに美しい。すでにこれは芸術である。これを味わわずに死ぬことは許されない。ということでドラキュラフェチを自認する私が徹底攻略で根性を叩き直す。初心者だからといって前半ごときであきらめるのは，あまりにもったいない。ブロック5以降でこそ，このドラキュラの真の姿が見えてくるのだ。この先を体験しないのはあまりにもったいない。たとえば，全裸で手招きしているパッキン美女と，ドラキュラのステージ5がセットされているX68000があったら，間違いなく後者に飛びつくほどだ（かもしれない）。冗談はさておき，本当に後半は素晴らしい。引きずりこまれたらもう絶対やめられない。朝の起きがけの布団ほどの重力だ。だからなにがあっても絶対そこまでたどり着け。

当たっても死なない。乗って越えよう

●ブロック1

ステージ1はいうことなし。まずは入り口のエフェクトでテンションを高めろ。ここで死ぬような者は，救いようなし。ただ，黒豹には注意。こいつは注意しないと見落としがちだ。突然突っ込まれるとあせる。よくこいつのことを覚えておこう。

ステージ2も問題ない。だが，初対面ではきっとダメージを食らうであろう目玉がいる。これも覚えておくように。こいつは，近づいてくるときに流し打ちで1発，そのあと余裕がなければ，まくってもう1発，ってのが基本。悲鳴がちょっとキュートでかわいそう。でも殺せ。

最初のエリアのメシはチェック。ここで誤ってサルまみれ（本当はセムシ男）にしてしまったときは，一度下に降りてから再び戻り，メシを食おう。基本的にはステージ内では画面が変わっても戻ってこられる。

各ブロックに隠された体力回復アイテム。その場所を探し出し戦いを有利に進めろ！

あとあと連射パネル稼ぎ，ハート稼ぎが必要な場面があるかもしれないので，覚えておこう。しかし，このサル穴は使える。下手で下手でしようがない人はこのサル穴で稼ぐ。上から下打ちで殺しまくればOK。美しくないヘボプレイの代表だから，美しき紳士，淑女には禁止。あとは半魚人。まあ落とされないようにって程度。後半には意味がない隠しキャラがある。

ステージ3はステージ1と変わりない。ここで出る斧は必要だ。ボスは斧がいちばん。打ったら分裂してから突っ込んできて，また合体の繰り返し。突っ込んでくる前に余裕を作っておいて，突っ込んできたら歩いて逃げるだけ。弱すぎて話にならぬ。

●ブロック2

いよいよここからが難しくなってくる。ステージ4はコウモリだらけ。いやらしいコイツらを味わえ。2周目以降は極悪だが，1周目ならまだカス。よく狙って正確に殺せるよう，しっかり慣れておこう。また細胞壁は触ってもダメージじゃない。あんまりかっこいいから切りたくないが。

鋭い人ならステージ5にルートが2つあることに気づく。上はあえて困難に身を委ねるM用。下はクロスもさっさと取って血にまみれて愉しむS用。

上はちょっと痛そうな針野郎と弾む目玉のダブル攻撃でリンチされる。ゆっくり進んで確実に殺していくのが正しい。

ぜひ一度は死んでほしいトラップ

マジシャンはできるかぎり逃げ場を残せ

下にはいろいろいる。最初の滝にクロスあり。重なって打てば落とさずに取れる。針野郎はあせらず撃退。ウツボは無敵アイテムを取ったあと突っ走れば平気。なくてもジャンプでかわせば自滅する。デモプレイがかっこよく決めているから参照すればいい。あとのウツボはムチ打ち4発で殺せる。飛び出たあと，ゆっくり戻るときに飛び越えても可。10点ももらえてラッキーってところだ。

合流するあたりにはアイテム野郎が隠れている。肉もいいがクロス，2連射なんかもお勧めできる。

さて，いよいよ前半最大のハマリ場，ステージ6だ。最初は絶対に死ぬ。でも根性で抜けてもらいたい。温泉にありがちなライオンを破壊するといよいよスタート。イカダは途中の島（？）に当たるごとに傾き，崩れてゆく。傾くと滑るから，落ちないよう中心をキープすべし。ただ，歩いて登るのはきついから，傾いたら即ジャンプ。半魚人は自分と高さが合ったとき以外は炎をはかない。これも基本的にはジャンプでかわすようにすること。パターンはないからアドリブで気合を入れて抜けてほしい。最後のほうでイカダは崩れ去り，島が連続しているところからは自分で進んでいく。途中にはクロスもあるが，無理は禁物。だましでナイフもあり。だまされるな。

そして，ここを抜けた努力を一瞬で無駄にする極悪なボスの登場だ。とにかくイカダの左端をキープ，攻撃よりもキープを優先させる。炎に当たれば即死亡と思え。これもあせらず，イカダが傾いたらジャンプして攻撃，を地道に保つ。クロスがあれば連射しろ。なにがあっても無理は禁物。

●ブロック3

覚えてないと結構はまる。でもせめてボスぐらいまではいきたい。

素人は，おそらくステージ7のかつてより忌み嫌われている，鳥＋サルのダブル攻撃で死ぬ。基本的にサルが落ちてくるタイミングに合わせ，速攻で殺すべし。タイミングが命だ。溜めるとかなりのハマリとなるが，殺せなかったときはあせらず冷静になれ。このサルは，正面を向いているときは近づいてシモンを飛び越え，後ろから追いかけてくるときは突っ込んでくる。このパターンを頭に叩き込んで，後ろから追われたときは，向き合うことを習慣づけろ。

イモ虫は起き上がって舌を出したときに，しゃがんで打たないかぎりは殺せない。そのほかは判定なし。あせらず近づいて殺そう。また，矢は打ち落とせる。かなり気持ちイイが遊びすぎないように。本体を倒すときは，少しずつ前進しながら殺すといいだろう。途中にはメシもあるから安心。

さて，続いてまたもや厳しいステージ8。ここでは，カエルがいやらしい。しかし，安全な陸地におびき出すと，彼らはジャンプできずに下でもがきだすので，そこを下打ちで撃退しろ。これで完璧。

小さな鳥はそれほど強くないから気にするな。あと，泥はかなり沈まないと死なないからそんなに恐れずに進め。途中，下を潜って取れるロウソクは薬草だ。全面中おそらくここにしかないが，アイテムならクロスを勧める。

ステージ9はちょっとてこずる。とにかく滑ることに注意。どうしてもその場で止まりたかったらジャンプ，またはしゃがめばOK。最初のほうのトラップ，せりあがる氷柱で一度は死ね。攻略ヒントは，戻ってくるときがゆっくりだということ。もしくはできるかぎり奥に乗れば，そのまま飛び越えられる。ウツボも似たようなもの。あとは，コウモリ，サルに注意すればいい。

さて，いよいよイカすマジシャンの登場だ。基本攻撃は3種類。青色のときは氷柱。別にダメージはなし。その後のためにできれば上に乗っておきたい。黄色はナイフ。とにかく避ける。1回転してから飛んでくるからタイミングを覚えよう。緑色は龍。がんばって避けるか打ち落とせ。食らうとハート，なければムチを失ってしまうから注意。ここでムチを失うと次の面のスタートがちょっとキツイ。そして，ある程度いたぶると，今度は魔法陣で悪魔召還を始める。悪魔は，まず真ん中で様子をうかがい，途中で左右のどちらかを向く。その反対に逃げれば平気。氷柱で逃げられなくなることもあるので，ちょっと注意しておこう。

●ブロック4

ステージ10は上に登っていく。ムチがなかったりしたら最初の骨竜はシカトしてもかまわない。メデューサがそろそろいやらしさ満点。落ち着いて確実にいこう。

その後は邪教の館とでもいうところか。マップをちゃんと見ていたあなたならこの最大の隠しに気づく。おいしいものがたくさんあるから絶対チェック。ここもそんなに問題ないだろう。何周目かでは血の鼻水があなたを襲うが……。

眠れぬ夜

さあ，ひととおり前半を貫いた。あとは自力で素晴らしい後半を進んでほしい。とにかくいままでのところはさっさと超えてほしいのだ。後半はまた次号に譲ることにする。

難しいといっても，まず操作を極めれば簡単。なによりしっかり覚えるのが正しい道だ。結局のところ覚えゲーだろうと思う。あまりに素敵な後半にたどり着いたら，もうあなたはやめられない。クリアするまで離れることは許されないのだ。眠るのを惜しみ，仕事はサボれ。あなたは，この素晴らしい一夜を過ごすために生を受けたといっても過言ではない。

めったに避けられない竜。ムチまで取られるな

とってもいやらしいメデューサ

怒涛の斬影拳に希望を見た！

Nishikawa Zenji
西川 善司

魔法株式会社第2弾として発売される「餓狼伝説」。たび重なる発売日の延期に，やきもきしているファンもいるだろうが，待たせせただけの出来に仕上がっているので，安心してほしい。

さて，予定日より大幅に遅れての発売となったこのゲーム，待ちこがれていた人も多いのではないかな。6月号で紹介したバージョンは，まだサンプル版であったため正確な仕様が報告できなかった。翌月にはホームデータからこのソフトの情報がこなくなり，本当に発売されるのか不安になってしまった。さらに，発売開発元のホームデータが突然の社名変更。イカツい都市感覚的な名前に変わるかと思えば「魔法株式会社」。

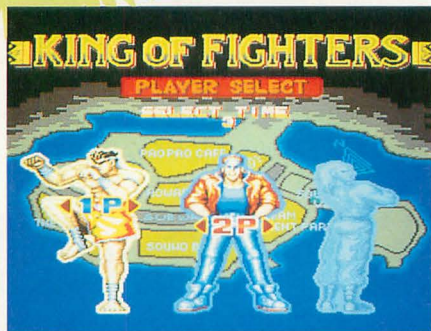
これを聞いたとき，私は「餓狼伝説」のX68000版はアダルト美少女ものになるのでは？ とか，ソフト購入者にもれなくヤギの血プレゼント！ みたいな変な特典がついてきたりしないだろうか，などと真剣に心配したり，社名変更で一層売れてしまったりして，などと無責任に想像をふくらませ，違った意味でこのソフトの発売を楽しみにしていた。

というわけで，それではまずいちばん知りたいと思われる，この「餓狼伝説」。X68000版の最終的な仕様をご報告しよう。

X68000版はこうなった！

●対戦モードはこうだ！

主人公キャラクターは，ご存じ「テリー・ボガード」「アンディ・ボガード」「ジョー・東」の3人。対戦モードではこの3人からの選択となる。オリジナルのNEO・GEO版

X68000用　5"2HD版4枚組8,800円（税別）
魔法株式会社　☎078(261)2790

操れるキャラクターは3人。誰を選ぶか？

と違い，同キャラの対戦も可能となっている。6月号の記事の同キャラ対戦の様子を示した画面写真を見て，同色のキャラクターが2人向かい合って戦っているのに気づいたと思う。「これじゃ接近戦になったときどちらがマイキャラなのか判別が難しくなるよ」と心配した君，安心したまえ。完成版では対照的にカラーリングされた2人が戦うことになる。

さらに，対戦マニアのために「対戦専用モード」が装備されている。このモードは，対戦モードにおいて決着がつくとすぐにタイトル画面に戻り，次の対戦を行えるというものだ。

そして，このゲームに関していちばん多かった質問「敵キャラを使用した対戦が行えるか」の答えは，残念ながらNo。メガドライブ版，スーパーファミコン版では敵キャラを使用できるだけにちょっと残念だ。カポエラのリチャード・マイヤーvs棒術使いのビリー・カーンのような，夢の対決が見られないのは実に残念だ。

●2人協力プレイはこうだ！

「対戦専用モード」でない通常モードでの2人プレイは協力プレイとなる。つまり，CPUが操るひとりの敵を2人でボコボコにしてしまうリンチモードとなるのだ。もちろん途中参加も可能。ただし，最終面ではストーリーの都合上，2人協力プレイはできなくなっている。

2人協力プレイで敵を倒したあとは，1Pvs2Pの対戦に一変し，この勝負の勝者が次のステージに進み，新たな敵と対戦できるというシステムになっている。

また，2人協力プレイでは，味方の攻撃がマイキャラにヒットしてしまい，接近戦になると，誰が誰を殴っているのかわからなくなり，ほとんどプロ野球の乱闘状態。よほど気持ちのわかりあった2人で遊ばないと，あとで「てめえ，このやろう！」と現実世界での対戦モードに陥る危険性もあるから気をつけよう。

●これに対応している！

「餓狼伝説」はもともと「パンチ」「キック」「投げ」の3ボタンゲーム。X68000のジョイスティックは，基本的に2ボタン仕様なので「投げ」を「パンチ」＋「キック」のボタン操作に割り当てている。しかし，電波新聞社製の3ボタンジョイスティック「XE-1ST2」にも対応しているので，これを使えばオリジナルの操作感覚でプレイすることができる。こういった配慮はうれしい。ぜひほかのソフトハウスさんもやってほしいものだ。

また，ハードディスクにも対応。「餓狼伝説」は大量のデータを頻繁に読むので，快適にゲームをするための必須アイテムといえる。専用のインストーラがついているのでハードディスク初心者も，安心して自分のシステムに組み込むことができるだろう。

●うちのマシンで動くかな?

X68030にも対応。しかし，030だからどう

最初の対戦相手を4人の中から選択

2人対戦モード。同キャラでも可だ

ライジングタックルでリチャードとシンクロナイズ

だ，というような特別仕様はない。

また，一部の雑誌広告や記事でメインメモリは4Mバイト必要であると報道されたが，通常のひとりプレイならばメインメモリ2Mバイトであっても動作する。が，2人協力プレイは，2Mバイトではプレイできない。一応公称4Mバイト必要ということになっている。こちらで調べてみたところ3Mバイトあれば動くようなので，ハードディスクにインストールした場合は，これを目安に起動するといいだろう。ちなみにHuman68kではコマンドラインから，

A>MEMFREE

でその時点のフリーエリアを確認できる。

BGMはMIDIにも対応

BGMはエスニックでユニークなものが多い。パオパオ・カフェのステージの音楽では，アラビアンなメロディに乗せてAD PCM音源のボーカルとコーラス隊が熱唱してしまう。一聴の価値ありだ。内蔵音源でもかなりのクオリティのBGMを聴かせてくれる。実は音楽，効果音制御にZ-MUSICを採用している。となると，もちろんBGMはMIDIにも対応。対応音源はSC-55/CM-300などのGS音源とMT-32系音源の2種類。

MIDI対応モード時はMIDI音源と内蔵音源とのアンサンブルになるので，かなりゴージャスなサウンドが響く。

移植完成度はいかほど?

さて，最も気になるこの項目。結果からいうと100点満点中80点はあげたい。CPUの思考ルーチンもほぼ再現されているし，キャラクターの動きもオリジナルそっくりだ。「投げ」ボタンの割り当てが変更されたこと以外，アーケードと同じ感覚でプレイできるだろう。スーパーファミコン版で省略されてしまった，奥と手前の2段階の奥行きによるゲーム性も健在だ。

背景動画のパターンや演出も一部省略されているようだが，こだわらなければ大した問題ではないだろう。マニアの視点から見るとオリジナルと違う点はもっとあるのかもしれないが，全体としてのまとまりは悪くない。

それにかなり大型のキャラクターを動かしているのに，ゲームスピードは10MHz機でも納得のいく操作性とレスポンスが保たれている。このあたりは「魔法株式会社」のアンビリーバブル・マジックといったところか(待つ，突っ込み)。

目立つ欠点をあげるとすれば，ディスクアクセスとディスク交換の多さだろう。画面が真っ暗のままディスクアクセスランプが点灯しっぱなし……という状態がゲーム中随所で見受けられる。しかし音楽，効果音，背景データやキャラクターのアニメパターンをすべてメインメモリ上に置くのは，無理があるからやむをえないか。ハードディスクには対応しているので，ハードディスクユーザーはインストールしてプレイすることをオススメする。しかし，某C社の人気格闘ゲームが移植されるとすると「餓狼伝説」以上にディスクアクセスの多さが予想される。もはやゲームをプレイするにもハードディスクは，必携の時代に突入したのかもしれない。

そのほか，画面のドット比の関係上，多少横長画面になってしまっていることに気づく。これはディスプレイの垂直振幅つまみをいじってお好みの画面サイズに変更すれば解決できる問題か。マニアはそのあたりにこだわってプレイするのもいいだろう。

「餓狼伝説」の起動テクニック

X68000版は前述のとおりハードディスクにインストール可能である。ということはゲーム起動をある程度，自分流にカスタマイズすることができるわけだ。さらに，Z-MUSICを使用していることで，音響関係の設定が柔軟に行える。このあたりのテクニックもいくつか紹介しておこう。

●キャッシュを組み込め！

ハードディスクにインストールしたとしてもアクセスの回数は多いので，フリーソフトやHuman68k ver.3.0のHDキャッシュを組み込んでから起動したほうがいい。キャッシュに割り当てる容量だが，メインメモリが最低3Mバイト以上残っていることを目安に設定しよう。

●PCM8.Xを組み込め！

せっかくのAD PCMボーカルつきのBGMも，格闘家たちの雄叫びや効果音でカットされてしまい，いまいち臨場感が湧かない。そこで，江藤啓氏作のPCM8.X(Oh!X1992年6月号の付録ディスクに収録)をZ-MUSIC常駐前に組み込んでおけば，最大8音のAD PCM音の多重再生が可能となる。BGMのAD PCMも途切れることなく演奏されるし，効果音も重なってバリバリボンバー状態だ。

ただし，PCM8.Xは非常にCPUパワーを消費するためX68000 XVI以上のマシンでないと苦しいかもしれない。

・餓狼伝説起動バッチファイル例

```
PCM8.X
ZMSC -T10 -O3 -B GAROU_MH.
ZPD -S GAROU_SD.ZMD
```

注）-O3は8声あるAD PCM音を楽演奏用に3声，効果音用に5声割り当てることを意味する

●RS-MIDI/POLYPHONにも対応できる

パソコン通信上で配布されているZ-MUSICには，RS-232Cポートを使用した

負けてボコボコのリチャード

MIDI出力をサポートしているバージョンもあるので，こちらを組み込んでゲームを起動すれば，MIDIボードがなくてもMIDI音源のBGM演奏が楽しめる。

また，ネオコンピュータシステムから販売中のサブボード「POLYPHON」に対応したZ-MUSICもあるので，こちらを使用すれば「POLYPHON」を使用したMIDI音源のBGM演奏が楽しめる。POLYPHON用のPCM8.Xは10MHz機でも高速に動作するので，これを使用すれば10MHz機でもAD PCM音源多重再生しまくりの「餓狼伝説」が楽しめる。

マイケルにはローキックと斬影拳だ

斬影拳でふっ飛ばされたタンとテリー(笑)

キング・オブ・ファイターズ

まず6月号の技の紹介に一部誤りがあったので，この場を借りて訂正しておく(キャラクターが右向きのとき)。

●アンディ・ボガード

☆斬影拳　↙→　B

溜める必要はなく，実はコマンド技。

☆飛翔拳　↓↙←　A

●ジョー・東

☆タイガーキック　↓↘→↗　B

さて，必殺技を中心にした紹介は6月号でやっているので，そっち系統の話は6月号を見てもらおう。今月はひとりプレイモードの攻略を交えた話をする。

主人公キャラを3人の中からひとり選び，次に最初の対戦相手を4人の中からひとり選択する。それ以後の対戦相手は自動的に決定される。

●リチャード・マイヤー(カポエラ)

彼が天井の鉄棒にぶら下がって足を回転させているときは，飛び道具およびほとんどの技が撃墜されてしまう。よって，攻撃を仕掛けるのは彼が飛び込んできたときだ。テリーならバーンナックル，アンディなら斬影拳，ジョーならタイガーキックを使え。

●タン・フー・ルー(中国拳法)

ある程度ダメージを受けると彼は巨人へ変身する。巨人になるとメチャクチャ強くなる。ラインを変えられた場合は飛び込まないほうが無難だ。飛び込んでもほとんどがプロペララリアットで落とされるのがオチだ。変身後は接近戦も避けよう。変身前にできるだけ多くの体力を奪い，変身後はあせらず，ひたすら隙をうかがう待ちの戦法をとろう。

●マイケル・マックス(ボクシング)

ジョー・東ならば一度タイガーキックなどで近づいたあと，スライディングの連打で勝ててしまう。アンディならばローキックと斬影拳の連続ワザで楽勝だ。

●ダック・キング(パンク)

彼のローリングアタックは(納得がいかないかもしれないが)，飛び道具をスリ抜けてくるので注意。生意気にもテリーのバーンナックル，アンディの斬影拳などをスライディングで蹴り返すという小癪な技も多用してくる。また，近づきすぎると投げられるのでなかなか手ごわい。

●ファー・ジャイ(ムエタイ)

ある程度ダメージを受けると酒を飲み始め，急にスピードアップする彼。ジョー・東のタイガーキックと同じ技を連発してくる。このタイガーキックがくるのを先読みして飛び道具で撃退するか，いったんガードして技が出終わってマイキャラと重なったところを投げてしまう，というのがベストだ。ジョー・東ならば，ジャイが酒を飲んでいるときに近づいていって爆烈拳を連発すれば楽勝。

●ライデン(レスリング)

とにかく強い。その巨体に似合わずめっぽう空中戦に強い。技を出すとその隙をついて飛び込んでくるので，うかつに攻撃できない。特にバーンナックル，斬影拳などの飛び込みワザは，転ばされたりガードされて投げられたりするので多用は禁物。対処方法としては，毒霧を吹き終わった瞬間に必殺ワザを叩き込むしかないだろう。

●ビリー・カーン(棒術使い)

空中戦および飛び込みワザはことごとく撃退される。間合をとって飛び道具を中心に戦いたい。彼が棒を落としたときは無防備になる。このときに飛び道具や爆烈拳で体力を削り取ってやろう。

●ギース・ハワード(ボス)

烈風拳という飛び道具を連射してくる。しかし，ときおり連射をやめるときがあるので，この隙をついて飛び込み系の必殺技をお見舞いしよう。

●基本戦略

飛び込み系の技で転ばせたあと，起き上がりを飛び道具で削り取るという戦法が基本となるだろう。また，飛び込み系の技で飛び込んでいき相手がガードをしているところを投げてしまう，という技も結構使える。カプコンの「ストリートファイターII」と違い，必殺ワザをいかにすぐ出せるかという点が勝負の決め手となるのだ。

スラッシュキックが美しいジョー・東

次はなんだろう

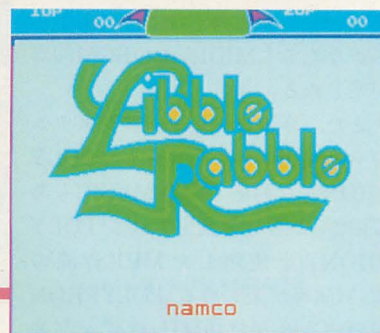
魔法株式会社では今回の「餓狼伝説」の反応しだいで，今後もNEO・GEO作品の移植を行うかもしれないそうだ。「餓狼伝説」を購入したらパッケージ同梱のハガキに，次の移植のリクエストを書いて送るといいかも。

しかしNEO・GEOは4ボタンが基本仕様だから，移植に関しての最大の問題は，ハードウェアの違いよりもジョイスティックのボタンの数だと私は思うんだな。たとえば「餓狼伝説2」は，絶対に4ボタンでないと特定の必殺ワザの操作ができないだろうし。そろそろX68000用の6ボタンジョイスティックが発売されないかねえ。6個もあればいろんなゲームに対応できるわけだし(意味深な発言)。

総合評価	0 — 5 — 10
ゲーム性	★★★★★★★
ゲームスピード	★★★★★★★★
操作性	★★★★★★★
グラフィック	★★★★★★★★★
サウンド	★★★★★★★★★

幻想と奇跡の世界へ

Yaegaki Nachi
八重垣 那智

「リブルラブル」は1983年にアーケードに登場し，なかば伝説化していたゲームだ。6月号ではサンプル版ということで，ルールを紹介するにとどまった。今月は実際のテクニックの紹介などをしつつ，内容の評価に入ろう。

奇跡！奇跡！奇跡！

このゲームで最も重要なことは，リブルとラブルを器用に操作する技術と，ゲームを有利に展開するための奇跡を確実に起こすための作戦という2つの点に集約される。前者はひたすら練習あるのみといわれているが，世の中にはテレビの「満点体操」のように，怪しげな動作を左右独立に行える人が多いようなので，最初はうまくいかなくてもそれほど悲観することはないだろう。私はパッドが苦手なので，標準添付のダブル十字パッドには馴染めなかったが，これも相性や練習の問題だと思われる。前例や類似品のない操作系なので，あきらめたりせずに，とことん練習してもらいたい。

最初は左右対称に囲むことから，しだいに不規則な形，宝箱の出し方，そして奇跡狙いの囲み込みというように，段階的に覚えていく方法が，面倒臭いかもしれないが，やはりベストだろう。

ある程度自由に囲めるようになったら，このゲーム最大のイベントである奇跡を狙いにいってみよう。奇跡を起こせば緑が甦り，ボーナス得点のチャンスもあるので，これを狙わぬ手はないだろう。奇跡を起こす，つまり宝箱を開き，中から飛び出す妖精のトプカプすべてを，効率よくバシシする方法について考えてみよう。

奇跡のために，まず宝箱の特性について理解することから始めよう。宝箱というのは，結局は1ドットの点である。そして，通常のフィールドには5×9で杭が並んでいる。この杭4本で囲まれた正方形を1ブロックとすると，4ブロック未満の面積でバシシすれば宝箱が出現する。4ブロック以上でバシシした場合は出現せず，囲んだ瞬間に光るだけになる。これは不規則な杭の面でも同じなので，体感的に面積がわかるようになると，スムーズに出せるようになるだろう。

肝心の奇跡は，画面中央上部の部分のキーワードがすべて揃ったときに起こる仕組みになっている。キーワードのそれぞれの文字は，宝箱が出現したときに飛び出すトプカプの色に対応していて，すべての色のトプカプをバシシしないかぎり奇跡は起こらない。トプカプは色によって逃げる方向が固定されており(図1)，一網打尽にするためには，あらかじめ宝箱を開けるときに囲みやすくしておく工夫が必要になってくるだろう。

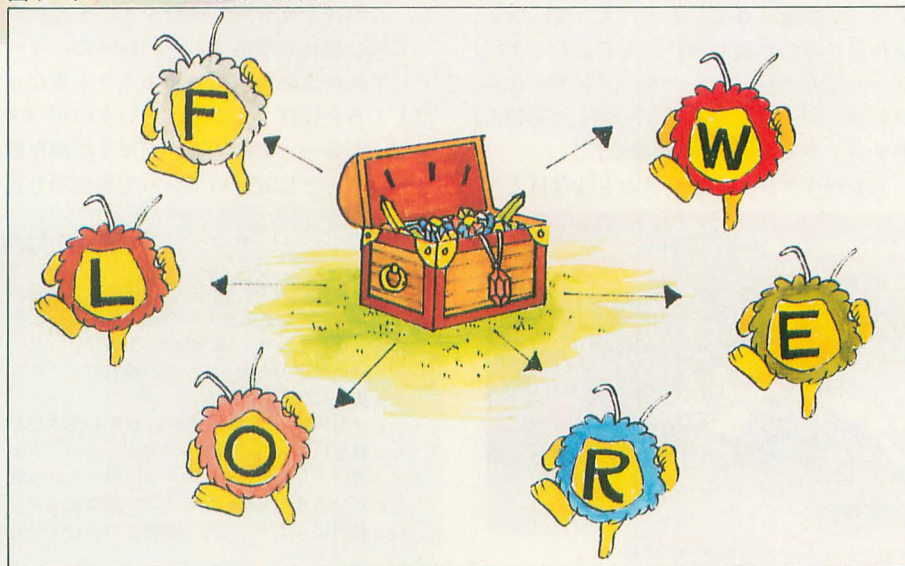
このへんのテクニックにはいくつかの基本形や流派(?)があるようだが，結局は3種類程度に分類される。確実に6匹を捕まえられさえすれば，特に具体的なやり方には制限はないので，自分でやりやすい方法をマスターするといいだろう(図2)。

つまり，宝箱が出た瞬間に一定以上のラインが確保されていれば，そのままで何匹かのトプカプを食い止められ，左右に逃げていく残りのトプカプを捕らえるのも容易だということである。宝箱の位置さえわかってしまえば，いくらでも奇跡が起こせるということになる。

使えるものはすべて利用

いくら囲み方に慣れたとしても，宝箱の位置がわからなければ，それこそ宝の持ち腐れである。1，2面では宝箱の位置が画面に直接表示されるが，それ以降は特殊な場合を除きノーヒントなので，宝箱の探し方もひとつの重要な作戦であることになる。

隠れている宝箱を探すには，もちろんその特性を利用すればいい。つまり，大きくバシシして光ったところで目星をつけて，少しずつ小さく囲んで探していけばいい。大きく囲むのは，2×2～3×3ブロック程度の範囲にとどめ，横幅に合わせて「下

図1　トプカプの色と逃げる方向

X68000用　5"2HD版　7,900円(税別)
電波新聞社　☎03(3445)6111

から」チェックしていくといい。

ここでなぜ「下から」なのかというと，これはリブルとラブルの特性のためである。バシシの効力は囲まれた面積だけでなく，瞬間的に外周やリブルとラブルの間のライン上にも効果が発生する。これは“待ち”を作ってバシシした場合に，予想が違って上をバシシしたときでも，下側の余ったラインが宝箱と重なっていれば，宝箱が開くことがあるからだ(図3)。こうなってしまったら，ラインの上下にトプカプが分散してしまうので，どんな名人でもお手上げである。

さらに宝箱を探す場合は，できるだけエネルギーをオーバーチャージして，無敵状態で探したほうが安全であるのはいうまでもない。

つまりは，植物の管理も奇跡を起こすための重要な要素ということになる。植物は最大で20個まで存在できるので，最初は育成に力を入れたほうがいい。

ある程度に成長させて農園状態になればしめたもの。紫の実が3個の種になることを考えれば，実が7個あれば次の世代でまた最大数になることがわかっているので，最大1世代あたり，実13個分のエネルギーが補給できる理論になる。そのあたりを頭に入れれば，無敵時間中にさらに補給するなどして，安全に宝箱を探し奇跡を起こすことは，それほど難しくないと言い切ってもいいだろう。

結局，「リブルラブル」はこうして頭と手の両方を最大限に駆使することで，楽しみが拡大していくゲームなのである。

移植の手触り

このように，オリジナルの攻略や作戦がそのまま通用することからわかるように，移植度は相当に高い。操作系の問題もオリジナルパッドでクリアしており，志の高い“ビデオゲームアンソロジー”のなかでも，1，2を争う出来だと思って間違いない。しかし，実際に基板と並べて比べると若干の違いはある。そういう意味では，やはりこれは移植なのである。

基板にMC68000が載っているので，メインルーチンなどはそのまま使われているような印象を受けるが，「リブルラブル」で使われているMC68000はグラフィック用のサブCPUにすぎず，実際のメインルーチンは6809で書かれている。当然，プログラムはそれらをもとに，X68000用にプログラミングし直されたものということになる。

オリジナルのプログラムを読み，各命令をX68000用に置き換えていくような方法をとれば，同じアルゴリズムのプログラムが出来上がるかもしれない。しかし，疑似乱数の偏りや，ハードウェアに依存したタイミングといったものがゲームの展開を左右していた場合，プログラムはそっくりでも，やってみると違うということが起こりうる。たとえば，この「リブルラブル」のオリジナルでは，キャラクターオーバーによる処理遅れを利用した攻略法があるが，X68000版では目立った処理遅れは発生しない(XVIでプレイしたせいも多分にあるだろうが)。これは別に間違った動作ではないのだが，やはり一部違うところがあるというチェックを受けてしまう。

植物の育成にいそしむ

中身を見てそれを写す方法は，確かに確実かもしれないが，それに溺れるあまり，手触りの違うものが出来上がってしまうこともある。内部でどのように動いているかよりも，やはり外から見た振る舞いが同じであるほうが，印象としてよく見えるのは事実ではないだろうか。結果的に違和感を感じなければ，方法などはどうでもいいことだからである。

まあ，全体的には絵も音楽もそっくりなので，オリジナルをよっぽどやり込んでなければわからないような違いなどは，無視してもいいのかもしれない。どちらかというと，中学・高校生の頃に夜12時過ぎても閉店しないゲームセンターで，せっせと奇跡を起こしていた世代よりも，幻のゲームとしてこのゲームの面白さを知りたがっているユーザーのほうが多いことだろう。

世代と時間を超えて愛されるゲームの領域に「リブルラブル」が到達し，これからもより多くのファンに支えられることを願いたいものである。

図2　宝箱の囲み方

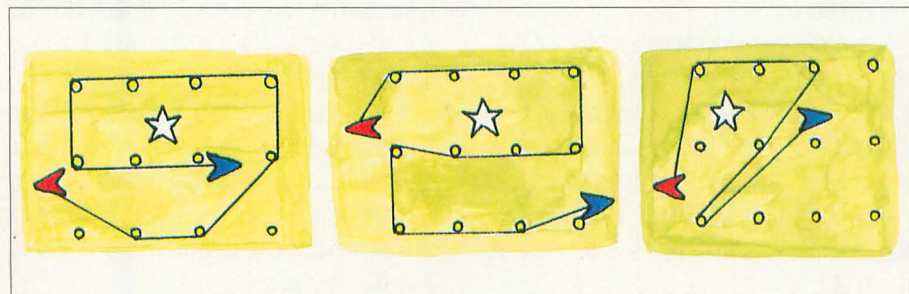
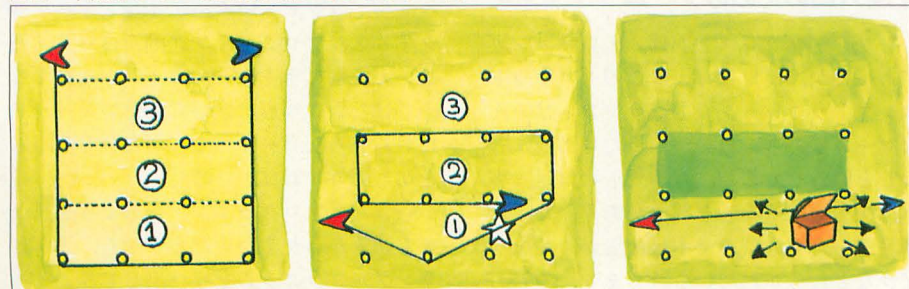



図3　下側のラインで宝箱が開いてしまうケース

時代の流れのなかで

最近のゲームセンターは大型テレビを使ったアップライト筐体ばかりですが，10年前はテーブル型の筐体が全盛で，レバーは逆手で持ってというのがスタイルだったわけです。そういう筐体でプレイする場合は，ボーナスステージで画面の上に宝箱の位置の目印を置くことができました。この目印用として配布されたオフィシャルアイテムが「バシシマーカー」なのですが，X68000用にもこれが付録についてきます。が，パソコンのディスプレイは立っており，使えないのが悲しいです。誰か，いい方法を考えてくれませんかねえ。あ，モニタを上に向けるというのは反則ですよ，うんうん。

総合評価	0　5　10
ゲーム性	★★★★★★★★
技術	★★★★★★★
サウンド	★★★★★★
グラフィック	★★★★★★
オマケ	★★★★★★★★
ノスタルジィ	★★★★★★★★★

THE SOFTOUCH

●ロボットコンストラクションR.C.

自家製ロボットを作る喜び

Shibata Atsushi
柴田 淳

コンピュータの中に自分だけのロボットを作る。好みのパーツを選んで筐体を組み上げ，それに適したプログラミングを施してやる。競技場で力を試したあとは勝っても負けても改良を重ね，最強のロボットを目指そう。

その昔，「パソコンが使える」という言葉は「パソコンが作れる」と同意語だった。自分ひとりで使える，比較的安価なコンピュータという意味でのパソコンはどこにも売っていなかったし，当然，「自分で作る」しかなかったのだ。

時代が少し進むと，そこそこのお金を出せば完成品のパソコンを手に入れられるという状況になり，コンピュータを使うために本体を「設計し組み立てる」必要はなくなった。しかし，それでもパソコンで走らせるソフトは圧倒的に不足していて，「パソコンが使える」ためには「ソフトが作れる」ことが必要条件だった。

現在，パソコンユーザーを自負する人に「ソフトを作れますか」と質問してみたらどうなるだろうか。首を縦に振る人は，おそらく10分の1にも満たないのではないか。「じゃあ，コンピュータの設計は」なんて聞こうものなら，状況がさらに悪くなるのは火を見るより明らかである。

便宜上，「設計できる」レベルのユーザーを第1次パソコン人口，「ソフトを作れる」ユーザーを第2次とし，第3次ユーザーは「ソフトを使うだけ」の人，ということにしよう。この3つのユーザーの現在の人口分布を見てみると，前述のとおり，第1次から第3次の順に，指数関数的に人口は増加していくことになる。

この分類というのは，お察しのとおり経済学の産業人口分類に倣っている。ちなみに，産業人口というのも3次まであり，1次は農/林/漁業，2次は製造，3次はサービス業に分類されている。また，1，2次の産業人口は生産人口などと呼ばれることもある。

X68000用　3.5/5"2HD版　8,800円(税別)
〈ブラザー工業(TAKERU)　価格未定〉
エレクトリックシープ　☎052(775)0530

経済学では，一般に経済が発展すると生産人口はサービス産業に流れていくといわれている。つまり「人口比率で見て非産的に」なっていくのである。で，ユーザー人口をこのように分類してみると，パソコンを取り巻く世界も，発展するにつれ非産的になっていくようである。

対決の前にはランキングや戦績が表示される

ただ，ユーザーが非産的になっていくことが悪ばかりか，というとそうではない。パソコンのように開発費のかかる工業産品は，販売台数が増えれば単価を安くすることができる。知識集約的な工業産品には，俗にいう「規模の経済」の原則が強く働くものなのだ。つまり工業文明であれパソコン界であれ，それを支えるのは基本的に人口なのである。

ところが，ここにX68000というパソコンが存在する。このパソコンは，人口分布で見ると生産人口の比率が異常に高く，いってみれば「発展途上国並み」なのだ。経済学でいえば，明らかに債務国に陥ってしまいそうな危うい状況なのに，超大国からの輸入にもほとんど頼らず，もう長いこと立派に自活している。

生産人口のためのゲーム

むかし『プラレス三四郎』というマンガがあったが，簡単にいえばこの「ロボットコンストラクション」というソフトはそんなゲームである。自分が作ったロボットどうしを戦わせるのだ。ロボットには足，胴体，腕という基本的な部分があり，たとえば足だったら人型の2本足だとか，キャタピラなどに分かれている。つまり，それらを組み合わせて，1体の完成されたロボットを作ることがゲームの出発点である。

いや，逆にいえば，これはあくまでも出発点でしかないのだ。というのは，パーツを組み合わせ，オリジナルのロボットを作っただけでは，ロボットは押し黙ったままで動いてくれない。

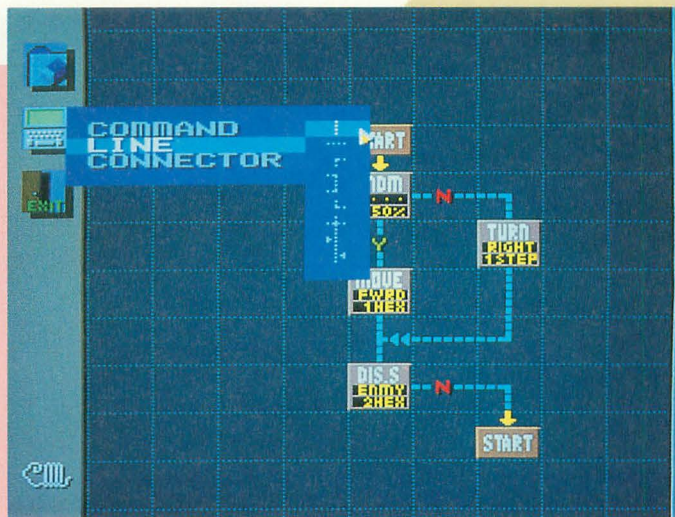
じゃあ，どうするか。ロボットはあらかじめ打ち込まれたプログラムをもとに，自立的に動作するのである。「生産人口」のためのゲームであるゆえんは，実はそこのところにあるのだ。

ロボットを動かすプログラムを作らなければならないとはいえ，インタフェイスは工夫されている。いちばん特徴的なのがこの部分であるが，ロボットを動かすためのプログラムは，CやBASICのようにテキストのソースではない。

自分のロボットを1体作ったとする。メニューでコンピュータのアイコンをクリックすると，プログラミング画面に移る。とはいえ，別にテキストエディタが起動するのではない。矩形に区切られたフィールドにいろいろなパーツ（これが命令となる）を切り張りすればいいのだ。

いきなりプログラミングをしようと思っても勝手がわからないだろうから，まずはサンプルプログラムを呼び出してみよう。

接近戦をアップにして撮影したもの

フローチャートは命令パーツをラインでつないで作る

画面写真を見てもらえばわかると思うが，命令となるパーツが，ちょうどフローチャートのように並んでいる。

さて，プログラムは必ずSTARTというパーツから始めなければならない。いってみれば，Cのmain関数みたいなものだ。ここから，次に実行する命令に向かって線を引く。つまり，START命令の下に縦線を張り付ければいい。

このゲームでは基本的に，このように命令と命令の間を線で結ぶことによって，プログラムの流れを決定する。鋭い読者はもうお察しのことだろうと思うが，命令のなかには当然条件分岐もあるから，分岐命令から分かれる線は成立，不成立の2つの線ということになる。

どこまでできるのか

「どんなプログラムが組めるか」という問題は，このシステムがもっている命令の種類によって決定される。ひと言でいえば，「そつなく無難な命令群」が用意されている，として差し障りはないだろう。

まず，ロボットを移動させる命令は当然あるし，そのほか装備している武器を使用するとか，指定位置にジャンプする命令などが筆頭である。これらの命令は抜け出る方向が1本ということになる。

次に，条件分岐命令がいくつかある。基本的に，この命令の種類と機能がプログラミングの自由度を決定するといっていいだろう。これは，残りエネルギー，ダメージなど自機の状態を調べる命令，そのほか，乱数によって分岐する命令，カウンタ値による分岐，敵機までの距離を調べる命令とか，自分から見た敵の存在する方向を調べるものなどがある。ただし，実際の戦闘はヘックスで区切られたマップ上で行われるので，最後の命令は「自機から引いた6方向の直線上のどれかに敵がいるかどうか」を調べる命令である。

すると最も素朴なメインプログラムは，まず敵のいる方向を探知し，その方向に向かって武器を使用，といった感じになるだろう。また，敵が見つからなければ，つまり，6方向の直線上にいなければ，適当に移動し，また最初に戻る，というふうになるだろうか。

しかし，より複雑なプログラムを組もうとするときには，命令よりも構造上の制約が問題になってくる。どういうことかというと，このシステムでは四角いマスに命令パーツをはめこむようになっているので，「ほかの命令との連結は最低4つ」という制約が課せられているのだ。

ある条件が成立するまで，同じ動作を繰り返す（たとえば，敵を見つけるまで左旋回する）プログラムを考えてみよう。敵の方向を検索する命令が先頭に来て，敵が見つかれば抜け，いなければ右旋回をする。右旋回をしたあとで，またもとの方向検索命令に戻れば効率的である。命令パーツを置ける領域は限られているので，同じ動作を実現するのであれば，より狭い領域に収まる流れ図のほうがいいのだ。

このようなループ構造では，まずこのループに入ってくる線を引く必要がある。次に2つの分岐で，先頭の分岐命令の周りは合計3方向つぶされる。最後に残った1方向はループを閉じるために使い，これでめでたしめでたし。と思ったら大間違い。先頭の分岐命令の隣が，ほかの命令や線でつぶされていたらどうか。

実際，少しでも複雑なプログラムを組むとなると，こういう場面に頻繁に出くわす。最終的な局面では，いつもこのような「構造上の制約」に悩まされることになる。まあ，基本的に縦積みの流れ図を作っておけば問題ないのだが。

このシステムにはCのSWITCH命令のような多重分岐命令は存在しないので，理論的にはどんな構造のフローチャートも書けることになる。つまり構造上の制約に悩まされるかどうかは，あなたの「プログラミングに対する美学」にかかっているのだ。

エラいロボットを作る

このゲームでは，あらかじめ30体のロボットが用意されている。それぞれに凝った

片方が負けた瞬間。左にあるのは地雷

各パーツは好きな部品が好きな色で

飛び道具は当たればいいが，そうでないと……

たくさんのロボットでトーナメント戦を行うことも可能

プログラミングがなされていて，当面はそれらのロボットを倒すような強いヤツをこしらえて，上位ランク入りを目指すのがゲームの目的となる。

では，強いロボットとはどんなロボットか。まず，装備の特性を生かしたプログラムを組むことが勝利への第一歩だ，ということがいえる。飛び道具を持っているのなら，離れたところから攻撃したい。飛び道具は威力がないので，ダメージの大きい格闘で敵をブチのめしたいという血の気の多い向きには，早めに敵に取りついて，一気に勝負を決めるプログラムがいいだろう。プログラミングレベルでは，敵の方向を探知して，そこから敵に近づくか，つかず離れずするかを選べばよい。

地形効果も考慮しなければならない。戦闘の行われるマップには3種類あり，障害物があるものとないものがある。障害物があるマップ上で，前者のような飛び道具を持ったロボットを戦わせるとどうなるか。

敵の方向というのは障害物には関係なくサーチされるので，周りに敵を見つけたとしても，せっかく撃った弾が障害物に当たってしまうことがある。飛び道具には弾数制限があるので，ムダ弾ばかり撃っていられない。

ところで，あらかじめディスクに入っている30体のうち，上位にランクされているロボットはどんな戦法をとってくるかというと，これがなんと，ほとんどが逃げ回るタイプなのだ(完成版では変更される可能性あり)。逃げ回って，敵のエネルギーが尽きるのを待ち，キツい一発を浴びせてくるヤツらが強いのである。

これらのロボットに打ち勝つためにまず考えられるのが，相手の索敵範囲外から飛び道具で攻撃するタイプのロボットをあてがうということだ。しかし，障害物の多い地形では先ほどいったようにムダ弾を撃つことになってしまい，いまいち効率が悪い。いろいろ試した結果，いちばん強いロボットというのは，逃げ回る相手を隅に追い詰め，接近戦で確実にブチのめすタイプのロボットのようだ。

そのほか，自分の周りに地雷を仕掛けて，あとはジッとしているタイプとか，それこそアイデア次第でいくらでもユニークなロボットが出来上がる。ただ，ユニークさと強さは必ずしも一致しない。ウケ狙いに走るか，強さを追求するか，これもまたあなたの選択次第である。

パソコンを使える人に

僕は，プログラミングを知らない人は基本的に正しいパソコンユーザーではないと思っている。しかし，僕がいう「プログラミングを知っている」というのは，「言語を使ってプログラムが書けないまでも，マクロやバッチは書ける」ユーザーを含む。バッチファイルが書けないとDOS上での作業効率はガタ落ちするし，表計算などでもマクロが書けるのと書けないのとでは，やはり効率に雲泥の差がある。この深い断崖の上にいるか下にいるかが，「パソコンを使えるか使えないか」を決める決定的な差になっていると思うからである。

僕の通っている学校にも，パソコンを持っているらしい教授が数人いるが，彼らは英語を扱うにもかかわらず，TeXの表現力の恩恵に与かれないでいる。専門的な音声記号などを「一太郎」で打ち出すことを考えると，彼らに対し同情の念を禁じえない。悲しいことに，それでも彼らは最先端の工業産品を使っている気でいるのである。

最後になんだかグチっぽくなってしまったが，このゲームは，生産的な，あるいはそれを目指すパソコンユーザーにはうってつけのゲームである。パソコンを正しく使える人が，いかにもそれらしく遊ぶことのできるソフトといえるのではないか。

コンテストが開かれるらしい

ロボットのプログラムを組まなければならないとはいっても，フローチャートを作っていくだけだから，それほど難しくはない。すんなり遊べる層としては，BASICをかじったことのあるユーザー以上となるだろうか。

本文中には書かなかったが，このゲームはハードディスクにインストールすることができる。プロテクトもあえてかけられていない。おまけにESCキーでコマンドに復帰できるのだ。気軽に起動し，気軽に抜けられるのがありがたい。

望むべくは，こういうゲームは身近に（あるいはパソコン通信のネットワーク上で）同じソフトを持っている人がいて，そいつの作ったロボットと自分のロボットを戦わせてみるというのが理想なのだろう。その理想郷へのアプローチとして，ユーザーが作ったプログラムを集めてコンテストを開くらしい。

こういうふうに入力情報が多いゲームというのは，その情報の調達に困ることがしばしばある。そこで思ったのだけど，このゲームはTAKERUでも販売されるのだから，そのTAKERUを使って頻繁に，ロボットのプログラムを送り出せばいいのではないか。値段を安く抑えれば，結構いけると思うのだが。

総合評価	0　　5　　10
丁寧なつくり	★★★★★★★★★★
起動復帰の快適性	★★★★★★★★
凝ったデザイン	★★★★★★★★★
作る楽しさ	★★★★★★★★

連続美少女アニメ劇場,はじまるよ

宝魔ハンター
ライム
Jewel Beam Hunter Lime
1993 SILENCE All rights reserved.

Takahashi Tetsushi
高橋 哲史

「宝魔ハンターライム」は「機甲装神ヴァルカイザー」と同じく，サイレンスが制作したアニメーションアドベンチャーゲーム。連続モノとして数回に分けて発売され，価格もかなり安く設定されている。はたしてお買い得か？

凸凹コンビの魔物退治

人間界と魔界が友好関係を保っていた時代，魔族は友好の証として魔宝玉と呼ばれる石を人間界に預けていた。だが，人間との友好をよしと思わぬ魔族の手により，魔宝玉は盗み出された……。

と，重々しいナレーションが終わると画面はいきなりのチェイスシーン！ 主人公であるバースが，魔物と空中でのバトルを繰り広げるアニメーションが表示されます。どーでもいいですが，私は最初バースのほうを魔物だと思ってしまいました。だって，魔物よりよっぽど魔物らしいカッコしてんだもの，この人。せめてセリフをしゃべるときに，口パクぐらいしてくれればわかりやすかったんですけどね。

バース「てめー，やっと見つけたぞ。おとなしく魔宝玉を渡せ！」
魔物「そうはいくか。人間のような下等生物と友好だなどというようなやつらには，この魔宝玉は渡さん」

魔宝玉を奪い取ろうとして，いきおい余って魔物をぶちのめしてしまったバース。魔物はみるみる地上へ落下，あっという間に見えなくなってしまいました。

「あ～あ，何やってんのよ」

気がつくと相棒のライムがいつの間にかバースの後ろに立っています。というより，空中だから「浮いて」います。

ライム「せっかく見つけたのに～。魔宝玉ばらばらにしてどーすんのよ。せっかく，これで帰れると思ったのに」

ライムと妖怪の奇妙な戦い

どうやら，魔宝玉にはこの世に存在するあらゆるものの恨みや怒りを吸収して妖怪化する力があるそうなのです。その妖怪が人間を襲い出したら友好どころの話じゃなくなるというわけ。

バース「じゃあ，早く地上の魔宝玉を回収しないと……」
ライム「そう，一刻も早くね」

ということで妖怪化した魔宝玉を取り戻すべく，地上に降り立った2人はさっそく調査を開始します。そう，ここから普通のアドベンチャーゲームが始まり，コマンド選択に入るのです。

アニメーションだ！

この「宝魔ハンターライム」はシリーズになっていて，次々に続編をリリースするというほとんどOAVノリのアラワザを行うようです。確かに遊んだ感触も，「ゲームをやった」というよりは，「天地無用」みたいなおきらくごくらくOAVを1本観たという感じですから，そういう販売形態が正解かもしれませんね。何本かに分けることで，単価も抑えられていますし。

適度に散りばめられたギャグやキャラクタのかわいさ，ほどよくばかばかしくてテンポのいい脚本なども，私は気に入ってしまいましたが，アニメーションのほうもなかなかの出来です。特にライムの変身シーンなんかは最大のウリといってもよく，それゆえかゲーム進行上何度も出てきます。

ただ，正味20分ほどでゲームは終わってしまうので，そのくらいの時間に1,500円投資するのは損だと思うような人は買う前に考えたほうがいいでしょう。まあ私個人としては，1,500円なら満足できるセンかなと思っています。

PC-9801からのベタ移植ながら，24kHzモードに対応していたりという努力のあとが見られるのでよしとしましょう。何度もいうようですが，肝心のアニメーション処理はかなり滑らかですしね。

X68000用 3.5/5"2HD版2枚組 1,500円(税込)
ブラザー工業(TAKERU) ☎052(824)2493

どっちが魔物かわからない

ヲの人は狂喜するかも

しかし，“バース”と“ライム”は“ら○ま”と“シャ○プー”だし，ライムの肩に乗ってるのは“ラムネのタマ○ュー”だし，出てくるモンスターも“スーパーヅ○ンの徳○”だしで，もう狙うとこ狙ってるなあというのがひしひしと伝わってきますね。あっ，しまった。こんな一般人にわからないような話をしてしまうような私こそ，ヲの人……。

総合評価	0 5 10
シナリオ	★★★★★★
アニメ処理	★★★★★★★★
OAVノリ	★★★★★★★★★
価格	★★★★★★★★

万馬券を当てて馬を買おう

Akikawa Ryou
秋川 涼

光栄から競馬シミュレーションが発売された。競走馬2頭と資金1億円を元手に，馬主としての人生を始める。単純にレースを楽しむこともできるけど，やっぱり個人牧場を作り，有名レースに出場できるような馬を生み出したい。

競馬をテーマにしたゲームは，いろいろある。そして，その内容はといえば，馬券を買って，競馬レースを楽しむというものがメインであった。しかし，この「Winning Post」は違う。ギャンブルとしての競馬ももちろん楽しむことができるが，中心テーマは馬主を演ずることなのだ。

ところで，「Winning Post」というタイトルを聞いて，「Whipping Post」を連想したのは僕ぐらいのものであろうか。「Whipping Post」，すなわちムチ打ち台。単に，オールマンブラザーズバンドの有名な曲にそういうのがあっただけだが，まあ，どちらもムチを打つ場所であることには違いない。ちょっとコジツケ。

走れ！ コップイッパイ

ゲームはどうせ現実ではないのだから，やることは変わったことのほうがいい。そういう意味では，Wingsに通うよりは，馬を所有して立派な競走馬に仕立てあげるという題材のほうが面白い。しかし，なにぶん馬主の基礎知識などは誰もがもっているものでないことは確かである。そのあたりはどう処理されているのだろうか。

まずは，秘書の有馬桜子というお嬢さんが画面に登場する。このお方は，2頭の競走馬，そして，1億円というビッグなプレゼントを恵んでくださるばかりか，身の周りの世話までしてくれるという。このご時世になんと親切なお方，ありがたや。

ま，ともかく，もらったものはオレのもの，名前をつけてやらなければならない。ファンタジーRPGでお馴染みの，名前をつける作業は人によってはなかなか面倒なことのようだが，このゲームにかぎってはどんな人でもパスできるのではないか。

その理由は，既成の名前が用意されているというありがちなことだけではない。競走馬の名前という特殊な事情が，楽しんで名前をつけることを可能にしているのだ。

もし，何も浮かばないとしたら，あたりを見回してみよう。あっ，コップがある。じゃあ，1頭目はコップイッパイという名前にしよう。次にクリップを見つけた。2頭目はトメノクリップだ。

と，自分の部屋にあるものを見ながら，適当に名前をつけたって（少なくとも自分では）それらしく思えるものだ。聞いたところによると，編集用語にもそれらしいものがあるらしい。シュッチョウコーセー，シャシンニューコー，カンリョウコピー，ゲンコウセキリョーなどがそうらしいが，なんのことだかはさっぱりわからない。なんとなく，後ろにいくにしたがって恐ろしげな名前のようにも思える……。

うちの子をよろしく

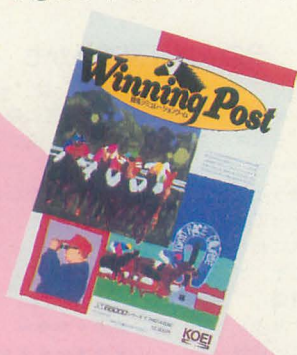
馬をもらったとはいえ，大事に家で育てるわけにはいかない。調教もしてもらわなきゃならん，ということで，どうやら調教師に預けるのが世の決まりになっているようだ。スタート時には拠点を東にするか西にするかを選択しているのだが，どちらにも15人，つまり全部で30人の調教師が存在する。このなかから，どの人にお願いするか。忙しい人には断られることもあるが，なるべくならやさしそうな人に預けたいなあ。ここは顔を見ながら，適当に選んでしまった。なかには「哭きの竜」に出てきそうな人もいる。

仔馬の命名や，活動拠点，調教師の選択以外にも，ゲーム開始前にすませなければならない作業は山ほどある。

自分（馬主）の名前に始まって，性別，年齢，血液型，勝負服のカラーリング，さらには呼ばれ方もいろいろと選べる。秋川社長とか，秋川の旦那とか。まあ，なんでもいいけど，女性の秘書に秋川の旦那と呼ばれると，なかなか面食らってしまう。

平日はお仕事

基本画面は平日と日曜とで分けられている。平日の画面には取引や駆け引きを中心にしたコマンドが並んでいて，いろいろな人との交渉がメインになる。

まずは，「情報」コマンド。ここでは所有馬やほかの競走馬，馬主，調教師，騎手，生産者，番組表を確認できる。次の「厩舎」コマンドでは，厩舎を訪れて情報収集，出馬登録，調教指示をする。「牧場」コマンド

X68000用3.5/5"2HD版4枚組12,800円（税別）
光栄　☎045(561)6861

厩舎を訪れて様子を見る

馬もいろいろパラメータをもっている

馬券を買うのはマークシートで

持ち馬がぶっちぎりで初戦に勝った

では専売牧場や個人牧場，育成牧場，スタッドを訪問し，情報収集や仔馬の取引。「馬主」コマンドでは，ほかの馬主と会って，仔馬や種株，肌馬の売買を行う。あとはシステム周りの環境変更を行ったりする「機能」コマンドや有馬桜子秘書にいろいろ聞く「秘書」コマンドがある。

最初のうちは「厩舎」コマンドと「秘書」コマンドがメインになるだろう。というのも，「厩舎」コマンドは自分の仔馬の様子を見るのに必要だし，「秘書」コマンドの有馬桜子さんはいろいろなことを教えてくれるからだ。

「秘書」コマンドのなかでもとりわけ役に立つのが，「競馬用語について尋ねる」というやつ。バリバリの競馬ファンなら無用の長物なのかもしれないが，初心者やちょっとかじった程度の者にとっては本当にありがたいコマンドだ。調べ方も簡単で，たとえば頭の1文字だけ指定すれば，その文字で始まる競馬用語がメニューウィンドウにズラズラと表示される。あとはそのなかから聞きたい用語を選択すればいい。

ふだん表示されるメッセージでも，尋ねることができる用語は色が変わっていてわかりやすい。ただし，プログラム中に収めたのはいいとしても，コマンドのひとつになっているので，いつでも使えるわけではないところは残念だ。

わからないことがあればこの人に聞こう

日曜日はいつも競馬

平日コマンドを終えると，日曜日がやってくる。日曜日は，そう，レースが開催される。ただし，「日曜画面」が表示されるのは，プレイヤーが「見る」レースがある場合のみである。「見る」レースは「機能」コマンドの環境設定で選択することができ，自分の持ち馬の出走レースだとか，GI競走，東西の重賞，特別戦，一般レースに関して，それぞれ見るかどうかを決定する。

日曜画面では平日画面のようにズラリとコマンドが並んでいることはない。ややこしいことは抜きねという感じで，競馬場がボンと真ん中に大きくあり，そのなかに掲示板，新聞売り場，オッズ(投票所)，パドック，そして，レース場がトップビューで描かれている。それぞれをクリックすれば，その場所に応じた行動がとれるわけで，持ち馬がレースに出るなら様子をうかがったり，関係のないレースでも馬券を買って儲けたりすればいい。最初から金は1億円もあることだし。

投票はマークシート(ややわかりにくいのが難点だが)，観戦はビジョンに映し出されるテレビ中継，と競馬場の雰囲気をうまく伝えようと努力したものになっているが，若干緊張感に欠ける気もする。実際に金を賭けているわけではないので，しかたがないかもしれない。

世界一の馬主

持ち馬をレースに出すには登録料が必要だし，そのほかにもいろいろと出費はある。それ以前に厩舎に馬を預けている以上，毎月末には預託料を払わなければいけない。この時点で赤字になるとゲームオーバーになるわけだが，最初から資金はわりと豊富だし，レースに勝てば賞金ももらえるので，ゲームを始めていきなりゲームオーバーということはまずない。

スタート直後は，出馬登録や調教指示といったことは調教師に任せておけばいいので(というか，経験値が上がらないとやらせてくんない)，何をすればいいのかわからないということもない。

もちろん，ゲームが進んで馬主として成長すれば，新しい仔馬を買ったり，牧場を作って仔馬を生産したりして，動くお金はより大きくなり，しなければならないことも増えていく。

そうして，個人牧場を経営しつつ持ち馬の育成にはげめば，必ずや最終目的であるフランス凱旋門賞に出馬できるだろう。そこで初めて，プレイヤーは世界一の馬主と呼ばれるようになるのだ。

馬は単なる資産ではない

作ったのは光栄，で，もちろんプロデューサーはシブサワ・コウだからして，だいたいのゲーム内容は想像するに難くないかもしれない。もちろん，パラメータは多いし，騎手や調教師などの登場人物の能力は重要な要素となっている。しかし，実際のレースや競走馬の育成に不確定な部分があるせいか，数値に気を配るようなことは極力抑えられている。

結局はお金をやりくりする経営シミュレーションにすぎないのだが，題材が題材だけに興味深くプレイできるようだ。

総合評価 (0–5–10)	
ゲーム性	★★★★★★
題材	★★★★★★★★★
競馬用語解説	★★★★★★★★

［第27回］
バグ取り

江口　響子

響子inCGわ～るど

自分がたくさん行進している
ひとり　抜けた
残りが　おまえはバグだといった
僕はバグなんだろうか？

徹夜つづきのバグ取りで，僕の頭はぼーっとしていた。なのに，脳の一部はハイになっていて，やたら冴えている。これがいわゆるトランス状態ってやつらしい。そんな意識のなかで，僕は自分が何人かに分裂したように感じたんだ。

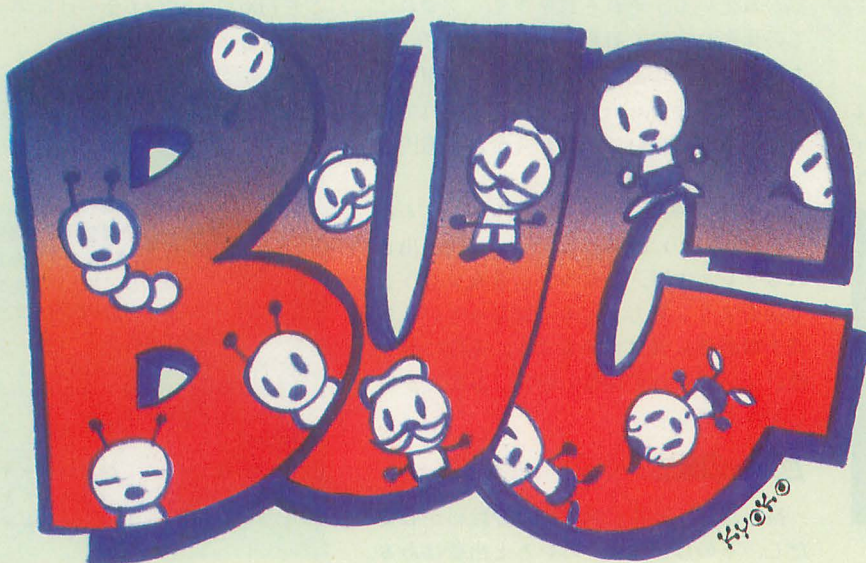
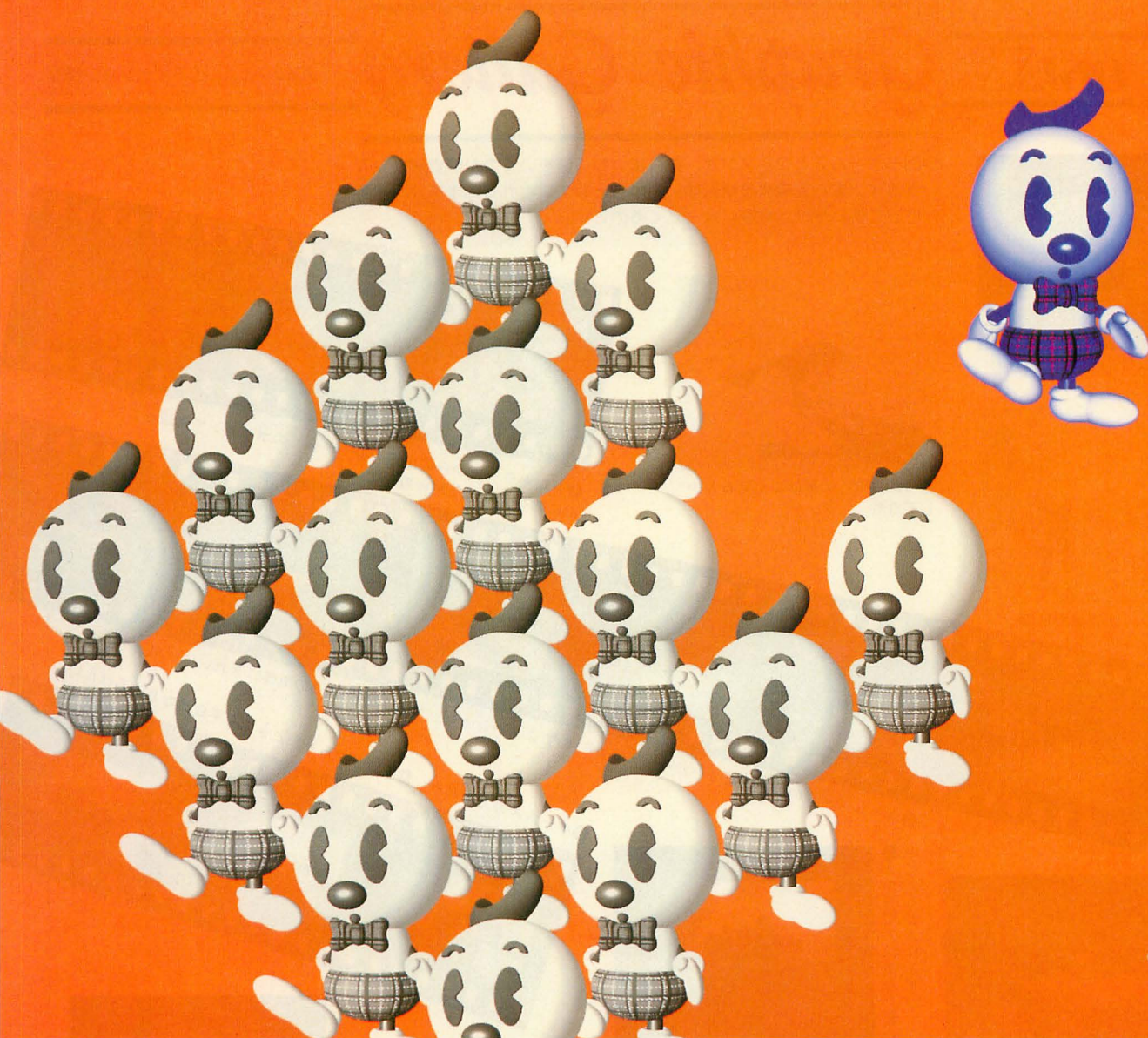
僕のことをちょっと紹介。Oh!X 4月号の「響子in CGわ～るど」に出ていたのが僕だ。あのときは中学1年生だった。5月，6月，7月と過ぎて，この8月号では，僕はある高校の1年生になっている。

時間の進み方が君ら現実の世界と違うからって責めないでほしい。ほら，マンガのなかだって，次のページでいきなり1年後になったりするじゃないか。あれと同じだよ。

話をもとに戻そう。いまの僕の仕事。バグ取りのアルバイト。ソフトウェアのプログラムミスを発見して修正し，商品として通用するものに仕立てあげる。持ち込まれるプログラムは他人の作ったものだ。見ただけでどんな構造か判断できる力がないと，この仕事はできない。また，忍耐力も必要なんだよ。バグのほとんどは，単純な入力ミスだからね。＋が－になっていたり，0と1が入れ替わっていたりする。それらをこつこつと直していくのさ。

この仕事をはじめるようになったのは，つい最近のこと。3年前にプログラマをやめて以来，コンピュータの仕事はしてなかったんだけどね。趣味で自分の作りたいソフトを作るのが，いちばん楽しいと思ったからさ。でも，高校に入ってホッとしたとたん，社会で自分の力がどれだけ通用するかを試してみたくなった。そんな頃，赤ら顔に口ヒゲを蓄えた社長が家にやってきたんだ。

「シロアリを駆除します。お宅はいかがですか？」
「あいにく，家の土台はコンクリートですので駆除はいらないんです」
「そうですか」

ヒゲ社長は，がっかりしてぽつりぽつりと話しだした。明治の頃から4代続いた，シロアリ駆除

の小さな会社を経営していること。最近は木造家屋がめっきり少なくなったせいで，仕事は減る一方なこと。すっぱりとシロアリ駆除はやめにして，新しい事業をはじめたいのだがそうもいかない。家訓があってそれには絶対に背けないのだと，社長はためいきをついた。

──生活を蝕む害虫を駆除し，世のため人のために尽くす──

というのがそれだ。

「そういえば，コンピュータの世界にもバグという虫がいて，そいつを退治するんですよ。バグ取りというのですが，なんだか似てますね」

「それだ！　バグだって害虫だ。ならバグ取りだって害虫駆除には変わりない。これで，立派に先代の遺志を継ぐことができる。ねえ君，一緒にやらないか」

こじつけくさいと僕は思ったけれど，妙な説得力があるもんだから，断りきれなかったんだ。その説得力を生かして，社長は営業を担当し，バグのほうは僕が駆除することになった。こうして僕は，ヒゲ社長と2人で仕事をはじめたのさ。

仕事は面白いように来た。商品化できないと思われたソフトが生き返る，と世間で評判になった。人に喜ばれるというのはいいもんだ。僕は世直しをしているような気分になった。仕事に夢中になるにつれ，学校がつまらなくなった。だんだん授業を休むようになった。

＊　　＊　　＊

きちんと高校と大学を出て平凡な社会人になるのが，僕には合ってると思っていた。いまでもたぶんそうさ。だけど一部分の僕が，こうじゃない，もっと違う生活があるはずだと耳元でささやく。僕は迷っている。ささやいている僕の一部はバグなのかどうか……。もしそうなら取り除かなくてはならないのかと……。

Oh!X Graphic Gallery

DōGA CGアニメーショ

今回もホームズとワトスンが，影にまつわる謎をいくつか解き明かしていきます。影を実現する手法はいろいろと考え出されていますが，状況に応じて使い分けが必要になってくるのです。

❶飛び跳ねる「HOUND」には映り込みや影が

❸「HOUND」の影はよく見ると少し不自然

❹有川キラー氏からの出題。UFOが多くのポリゴンで形成された山に影を落としながら飛んでいく

❷「SWORD2」の影は円形の黒いポリゴン

❺ホームズからの問題はピットから出ていくF1のアニメーション。ピットの影がF1の車体にきれいにかかっている

❽ホームズの方法なら，影のほうから見たときにも不自然なところはない

❻ワトスンの推理では境界面に半透明の板を置く

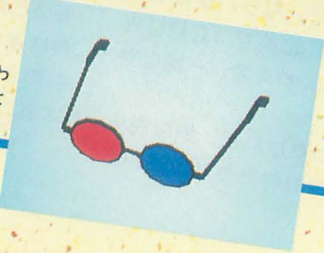
❼この方法では視点が影の中にあるときにまずい

❾❿⓫全体が明るい画像と一部をマスクした暗い画像を合成する。マスクは特殊な色を使って，あとで透明色に変換する

⓬サンプルにするためにモデリングした眼鏡

⓭タイル状の床の上で眼鏡が回転するアニメーション。眼鏡の影がほぼ完璧に再現されている

⓮白と重ね合わせれば透明色と黒の区別がつく

⓯眼鏡を置く床。模様があっても大丈夫

[特別企画]

夏真っ盛り，アマチュアリズムのX68000

夏！　開放的な夏！　熱暴走の夏！
突き刺さる日差しを避けて，コンピュータと戯れる
静かな部屋で行き詰まってしまったら脳ミソが沸騰するくらいに外で遊ぶ
徹底的に好きなだけ，遊んでしまおう
いまは夏真っ盛り，熱血コンピュータライフを楽しんじゃえ！

唸るマウス，描き込まれるスクリーン

ある電脳絵師のひとり言

Kawahara Youi 川原 由唯

本誌でいつも美しいグラフィックを披露してくれる川原氏。その実力は読者の皆さんも認めるところでしょう。今回は，どのような過程を経て作品が完成されていくのか，じっくり見ていくことにします。

尊敬している青木光恵先生の単行本をやっと手に入れてハッピーなういちゃんだす。夏には都築先生の画集(CG集)も出るそうだし，わくわくがいっぱいだなぁ。なっつは，恋の季節♪と(なんやそれ)。

さてさて，CG描いてる人の参考になるようなことって，なんかあったかな？

原稿依頼，受けてはみたけど，考えてみると人に教えるようなことしてないよな。教えてもらいたいくらいだし。

ま，いっか，手順とツール書いてお茶を濁しておこう(笑)。でも，普段，完成したCGを見ることはあっても，作業中の中間ファイルなんかは，見ることはないと思うのでそれなりに参考になるかな。というわけで，はじまりはじまり。

☆STEP1

ケント紙にロットリングかつけペンで下描きを描きます。あまり細かく描いても取り込み時にこぼれるので，適当に省略。このクセのせいで，最近細かい絵が描けなくなってしまった。昔は人に拒絶されるほどちまちました絵を描いてたのに……。

もちろん下描きにはトーンを貼ったりはしません。修正できなくなるから。基本的にセル原画調の絵を意識して描きます。つまりタッチなどを少なく均一な線で描くのです。

もっとも僕の場合，鉛筆画を取り込んで下描きに使ってしまったりすることも多々あるので，このへんはほとんど気分です。

参考にするのはもっぱらアイドルちゃんの写真とかそんなんばっか(笑)。雑誌とか広告の写真を見ていて，あ，いいなあと思ったものを参考にして描くことが多いですね。好きな映像作品からインスパイアされることもあるけど，思いどおりに描けたことなんて一度もない。うーむ。

☆STEP2

スキャナで取り込みます。ハンディスキャナの場合は原稿のサイズを小さくするために縮小コピーを使ったりもします。カラースキャナの場合は白黒，もしくは2値で取り込みます。「ハンディスキャナなんかじゃまともな絵にならないよ」とか，よくハンディを馬鹿にしたようなことをいう人もいますが，CGイラストの下描き取り込みに使う分には，十分だと思いますよ。

☆STEP3

取り込んだ絵を修正します。これは主に輪郭線のクリーンアップですね。2値で取り込んだときなどは，この過程の善し悪しが作品を決めるともいえるのでは。最も苦痛な作業ともいえます(人によるけど)。直線の部分は直線に，曲線は美しい曲線に，この時点で整えておくと，あとあと楽です。そしてベタペイント，ここではあまり細かいことは気にせずに，輪郭線の修正の一環のようなつもりで塗り塗りします。輪郭線が綺麗につながってないと色が流れ出るので，流れ出ないようになるまで輪郭線の修正を繰り返します。白黒多階調で取り込んでいる場合は輪郭線との境界が汚くなってしまうんだけど，まだここでは気にしません。全体に色がいき渡ったら，本当にこの配色でいいかをもう一度考えます。気に入らなかったらどんどん色を塗り変えます。

☆STEP4

さて，ここでいままでの絵から離れて，風景写真などを参考にしながら背景を描きます。僕の場合，背景はいつも適当なので下描きなしで直接描き出すことが多いですね。グラデーション塗りしたファウンデーションに，ランダムフラクタルなどのエフェクトをかけたりしながら，マウスやタブレットでがりがりと色を乗せ，ぼかしや色変換なんかを繰り返して気がすむまで手を入れます。ときどきこの背景のほうが気に入っちゃって，そのまま作品にしてしまいたくなってしまうんだよね(笑)。背景画は手描きのタッチが出せるので，MATIERがいいでしょう。

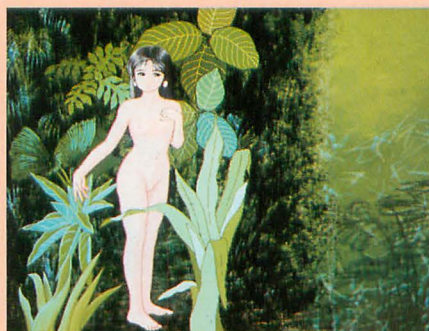
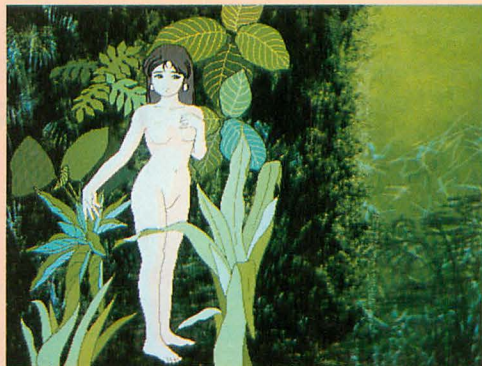
☆STEP5

背景が描けたらキャラクターと重ね合わせます。ブルーバック合成，要するにマスク合成ですね。輪郭線などの細かい部分の修正，目の周りなどはこの段階で手動でアンチエリアス処理を行ってしまう場合もあります。髪の毛の処理は，薄めにした極細ペンなどで，髪の流れ方向になぞるようにして描画しています。背景の葉の部分なんかも，陰影を考えて薄めのペンでタッチを入れていくのです。

そして，背景の葉をさらに細かく描画。これは，薄めたブラシで陰影をつけています。羽の線画を重ね，光度を上げるペンでなぞってタッチを入れ，さらに羽にタッチを入れて立体感＆透明感を出すように見せているのです。

んでもって，輝度を上げるペンをスプラッタブラシに設定し，適当なところにフラッシュを入れてさらに修正。また，さらに細かいブラシでホワイトのフラッシュを入れていきます。

これでだいたいの画面構成ができました。

最後に，肌の明暗（トーン）を輝度を上げるペンでハイライトを入れてから，ボカシペンで滑らかにします。

☆STEP6

あとは輪郭線の汚いところをルーペで拡大しながら修正していきます。極細のペンを薄めに設定して，こするようにして描くとアンチエリアシングのようなことができます。ただこのやり方だと，いまいちボケた絵になってしまうので注意しましょう。根気がいりますが，このあたりの作業も完成度にものすごく影響してきます。

☆STEP7

最後に仕上げです。ハイライトや影を要所要所に描き込んだりして，雰囲気を出します。強めにタッチをつけたり，レタリングなんかもたいてい仕上げのときに入れますね。レイアウトが気に入らないときは，縮小して調整します。

めざせ！ 日曜画家

ツールは，MATIERとZ'sSTAFF PRO-68Kを行ったり来たりします。１枚の絵を描くのに，それぞれ20，30回は立ち上げ直すんじゃないかな。画像の合成などは外部の小さなツール（フリーウェアや自家製）を使ったりもしますね。

しかし，近ごろ本業が忙しくって，あんまし絵を描いてない。道具にお金かかってるから，ちゃんと使ってあげたいのにね。日曜画家といっても最近は日曜も出勤してたりするからな。なんとかしてよ，もう。

下描きをスキャナで読み込んだところ

若干トーンを落として輪郭線をグレーにし，セル画調に色を乗せる

輪郭線周辺を，ルーペと薄めたロットリングペンで修正

おんまさん

背景は，まず雲の概観を描き始め，陰影を入れる。そして，にじみペンでこすり，筆のタッチを消してぼかす。最終的に細めのペンを使って，ハイライトを入れてみる

ひまわり子ちゃん

とりあえず，背景画とマスク合成したもの

髪を描く。薄めた太いペンで，だいたいの明暗を描く

背景と人物の境界線付近に浮き出た塗り残しの白を，薄めたペンでこすって目立たなくする

薄めた極細ペンで，髪を描き込んでいく。MATIERのスーパールーペモードで描くと美しく仕上がる？

さらに髪に手を加える。明るめの色でつやを描き込む

ファインモードで縮小。アラを目立たなくする意味もありますが，ほとんど気分ですね

もともとの背景と重ねて，あとはほとんど微調整。目や輪郭などで汚く見えるところをルーペを使って修正していく

6畳一間のクリエイター

いつかその曲をつくる日まで

Takahashi Tetsushi 高橋 哲史

自称漫画家，趣味としてコンピュータでも曲作りを精力的に続ける，根っからのクリエイターである高橋氏。6畳一間で作られる作品は，どのようにして完成されていくのでしょうか。

X月X日（月）

編集の(J)氏から本記事の依頼を受ける。どんなことをやろうかと悩んでいたけど，今回は，最近ハマっている曲作りについてちょっと書いてみよう。

さて，やりたいことを決めたあとは，どんな曲を書こうかと帰りの地下鉄でぼんやりと考える。ガタンゴトンガタンゴトン……。「はっ，これだっ！」。突如単調な終電のリズムに触発された私は「無機的で単調だけどかっこいい曲を書こう」と，とんでもなく抽象的な目標を打ち立てる。具体的なことはなにひとつ決まってないのに，ふくらむイメージにもうわくわくする私。こうなると周りの視線も全然気にならない。あーでもないこーでもないと頭の中のメロディを口ずさみながら，イメージを固めていく。

この段階が制限なしで創作を楽しめる，いちばん幸せな時間といえるだろう。なにしろ頭の中で鳴っているのは，いままで聴いたこともないような素晴らしい曲なのだ。もちろんそこには機材の制約や自分の実力的限界もいっさい考慮されてはいない。ただただ，素晴らしいイメージのみに陶酔し続けるのだ。

まあ，作り始めなんてのはたいがいそんなもんである。とりあえず，この日は帰宅してからクラフトワークとP-MODELとYMOに浸って眠る。物理的進行いっさいなし。

X月X日（火）

昨日のイメージをさっそく形にすべく，起きてから3分でRCD.Xを組み込んでSTED2.Xを立ち上げる[1]。最終的には内蔵音源のみで作らなければならないのだが，とりあえずSC-55で骨組みを作って全体を眺めることに決めた。

さて，まずはリズムからいってみようかな。ガタンゴトンガタンゴトン……と。お，Gt.FretNoiseを低く使うとなんか変な味が出てきて面白いぞ。よし，ついでだからGunShotと混ぜてやれ。んー，いい感じ。こうきたらバックはEchoDropsしかねえよなあ。よしそしたら次は……。

9時間後。夕方に起きたのでもう深夜になっている。途中でお隣さんが帰ってきた関係上，モニタをラジカセからヘッドホンに変更したので（うちは壁が薄いのだ）いいかげん耳が痛くなってきた。とりあえず面白い感じには仕上がったのだが，ここで大きな問題が。

「こんなの内蔵音源にコンバートなんてできない！」

あまりにもノリ過ぎて，SC-55のパーシャル切れを心配しなければならないほど音を詰め込んでしまったこの曲は，とてもゴージャス（死語）になった反面，FM8音にはとうてい入りきらないほど太ってしまったのであった。私に進藤君ほどの音色作りと打ち込みの才能があれば，内蔵音源での再現も可能かもしれないが，残念ながら私にはそんな才能はないのだ。素直にあきらめて「この曲はこの曲でなにかに使えるでしょ」ということに決め，ストックディレクトリに放り込む。またいちからやり直し。

といっても今日1日の作業がまったく無駄だったかといえばそうでもない。9時間ずーっとSC-55と格闘していたおかげで，パーシャルリザーブの正しい使い方やコーラスや，リバーブの細かい設定などがわかったからだ。実は今日初めてマニュアルをきちんと読んだのだが（笑）。

私は必要に迫られるまで勉強をしないタチなので，こういった機会がないと基本的なことさえ知ろうとしない。自分のしたいことができる環境さえあればそれ以上は望まない人なのだ。これは私の基本姿勢で，「使いもしないことを勉強したってしょうがないじゃん」という信念に基づくものだ。マニュアルと多機能に振り回されるよりは，自分の知っている知識の組み合わせで済む場合はそれで済ます，ということが重要だと思ってるわけ。ただ機能（あるいは環境）に振り回される人が多い世の中に，ちょっと反発を感じてしまうんだよね。はい。

しかしあまりに集中しすぎて頭がふらふらになってしまったので，今日はこれでおしまい。朝日を眺めながら床に就く。ぐう

▼リスト2

```
 1: (i)
 2:
 3: (b0)
 4:
 5: (m1,4000)(aFm1,1)
 6: (m2,4000)(aFm2,2)
 7: (m3,4000)(aFm3,3)
 8: (m4,4000)(aFm4,4)
 9: (m5,4000)(aFm5,5)
10: (m6,4000)(aFm6,6)
11: (m7,4000)(aFm7,7)
12: (m8,4000)(aFm8,8)
13:
14:
15: /* chord  A(じわーっとした音)
16: (v1,0,59,15,0,0,0,0,0,0,0,3,0,
17:          31,0,3,3,0,27,1,4,0,0,0,
18:          31,0,0,0,0,23,0,2,0,0,0,
19:          31,0,3,3,0,26,1,4,0,0,0,
20:          20,0,0,6,0,8,0,2,0,0,0)
21: /* chord  b
22: (v2,0,33,15,0,0,0,0,0,0,0,3,0,
23:          31,15,5,3,0,26,0,4,0,0,0,
24:          25,0,0,2,0,40,0,2,0,0,0,
25:          31,0,3,3,0,32,0,1,0,0,0,
26:          25,0,0,5,0,8,0,2,0,0,0)
27: /* ごぉーんって音
28: (v3,0,4,15,0,0,1,5,0,0,0,3,0,
29:          31,1,1,1,0,3,0,1,1,0,0,
```

ぐう。すやすや。

X月X日（水）

今日は一所懸命漫画を描いていた。ということで曲作りは1日お休み。

X月XX日(木)+XX日(金)

両日ともに5時間ほどキーボードの前に座っていろいろと弾きまくるが，さっぱりよいフレーズが出てこない。ひとりで「あー」とか「うー」とかうなりながら苦しむ。こういうときはもうどうしようもない。だいたい曲なんてやつは，出てこないときは出てこないものなのだ。いわゆるインスピレーション待ち。これが結構辛い。

以前，同様の状態に陥ったとき，「ひらめきだけじゃやってけないから，理論のほうを固めてそれを土台にコンスタントに曲を書けるようにしてみよう」と思ったことがあった。で，楽典やらコード理論書なんかを買い込んで，いろいろやってみたのだが……。確かに間違いのない曲は量産できるようになった。

しかし自分で聴いてもさっぱり面白くない曲ばかりになってしまったのだ。そういう曲ばっかり作っているとだんだんいやになってきて，インスピレーションを待ってイライラしてるとき以上に，ストレスが溜まってしまったのだった。

確かに理論は必要だ。しかしそれはあくまで感性の補助にとどめるべきで，それが主体となっては本末転倒になってしまう。と，私は思っている。

X月XX日（土）

やっとときめきを感じられるフレーズにめぐりあえた。リスト1がそれだ。これを弾いているときやっと「ああ，これで1曲できるかな」と思えたので，さっそく作業開始。以前の間違いを再び起こさないように，今度は直接FM音源用で打ち込むことにする。とりあえず熱が冷めないうちにSUPERED[2)]を立ち上げて打ち込みを開始。パ

▼リスト1

```
 1: (i)
 2:
 3: (b0)
 4:
 5: (m1,4000)(aFm1,1)
 6: /* trombone
 7: (v5,0,45,15,0,0,0,0,0,0,0,
 8:   3,0,16,2,0,8,2,14,1,2,1,0,
 9:   0,18,4,0,7,1,5,1,4,4,0,0,
10:   18,4,2,8,2,5,1,4,4,0,0,25,
11:   4,0,7,2,0,1,2,5,0,0)
12:
13: (t1) @5t88b8<d16>a8b16g8a16f+8g16e8d8
14: (t1) e8._2(e 16 <e)&e2.>~2
15:
16: (p)
```

```
 30:          31,2,1,8,0,15,3,2,4,0,0,
 31:          31,1,1,1,0,4,0,1,2,0,0,
 32:          25,1,2,6,0,9,0,4,7,0,0)
 33: /* ごぉーんって音その2
 34: (v4,0,4,15,0,0,1,5,0,0,0,3,0,
 35:          31,1,1,1,0,3,0,1,1,0,0,
 36:          31,2,1,8,0,15,3,8,4,0,0,
 37:          31,1,1,1,0,4,0,1,2,0,0,
 38:          25,1,2,6,0,9,0,15,7,0,0)
 39: /* trombone
 40: (v5,0,45,15,0,0,0,0,0,0,0,3,0,
 41:          16,2,0,8,2,14,1,2,1,0,0,
 42:          18,4,0,7,1,5,1,4,4,0,0,
 43:          18,4,2,8,2,5,1,4,4,0,0,
 44:          25,4,0,7,2,0,1,2,5,0,0)
 45: /* ぎざった音
 46: (v6,0,48,15,0,0,0,0,0,0,0,3,0,
 47:          31,4,5,13,15,26,0,3,3,0,0,
 48:          29,6,1,12,12,21,0,1,7,0,0,
 49:          31,2,1,13,12,27,0,3,3,0,0,
 50:          31,1,3,9,8,1,3,1,0,0,0)
 51: /* bass
 52: (v7,0,43,15,0,0,0,0,0,0,0,3,0,
 53:          28,10,0,8,8,38,1,7,3,1,0,
 54:          31,11,0,9,8,30,1,0,7,0,0,
 55:          31,13,1,8,4,14,0,0,7,0,0,
 56:          30,5,5,10,2,0,0,0,3,0,0)
 57: /*
 58: (v8,0,58,15,0,0,0,0,0,0,0,3,0,
 59:          23,10,0,5,1,28,0,1,1,0,0,
 60:          23,11,0,8,5,16,1,2,1,0,0,
 61:          31,13,0,6,2,37,0,4,0,0,0,
 62:          22,4,0,5,0,0,1,2,1,0,0)
 63: /*
 64: (v9,0,59,15,0,0,0,0,0,0,0,3,0,
 65:          31,0,3,3,0,27,1,4,0,0,0,
 66:          31,0,0,0,0,25,0,2,0,0,0,
 67:          31,0,3,3,0,20,1,4,0,0,0,
 68:          20,0,0,6,0,0,0,2,0,0,0)
 69: /* trumpet
 70: (v10,0,42,15,0,0,0,0,0,0,0,3,0,
 71:          16,15,0,8,1,13,0,1,7,0,0,
 72:          16,12,0,4,1,55,0,4,0,2,0,
 73:          18,0,0,4,0,47,0,1,0,0,0,
 74:          18,0,0,10,0,0,0,2,0,0,0)
 75: /*ざざぁー
 76: (v11,0,41,15,0,0,0,0,0,0,0,3,0,
 77:          31,12,1,1,0,0,0,3,1,1,1,
 78:          31,0,0,0,2,11,1,2,0,3,0,
 79:          30,0,0,0,0,0,0,1,3,3,0,5,
 80:          12,1,4,0,0,0,0,0,0,0)
 81: /*B_D
 82: (v12,0,21,15,0,0,0,0,0,0,0,3,0,
 83:          31,30,25,14,1,0,0,0,0,1,0,
 84:          31,12,11,15,15,0,0,0,0,0,0,
 85:          31,12,11,15,15,0,0,0,0,0,0,
 86:          31,12,11,15,15,0,0,0,0,0,0)
 87: /*S_D
 88: (v13,0,60,15,0,0,0,0,0,0,0,3,0,
 89:          31,0,0,15,0,0,0,15,4,0,0,
 90:          31,19,14,15,3,0,0,10,4,0,0,
 91:          31,21,0,15,11,0,0,2,1,0,0,
 92:          31,17,17,15,2,0,0,0,255,0,0)
 93: /*E_D
 94: (v14,0,59,15,0,0,0,0,0,0,0,3,0,
 95:          31,4,0,1,0,0,1,13,0,1,0,
 96:          31,21,4,1,6,16,1,0,7,0,0,
 97:          31,26,0,15,15,0,1,9,0,1,0,
 98:          31,11,0,7,15,0,1,0,0,0,0)
 99:
100: /*=============================================
101: /*melody
102: /*INTRO
103: (t1) t8818o4@2@k-3v14@m4@s2        /*0
104: (t1) |: r4g2.f+1                    /*1,2 & 9,10
105: (t1) r4g2&g8a16g16f+2.f+f+          /*3,4 & 11,12
106: (t1) e2&e8ede                       /*5   & 13
107: (t1) t87f+2.f+f+                    /*6   & 14
108: (t1) t85e2&e8ede                    /*7   & 15
109: (t1) t82g2t76f+2t88                 /*8   & 16
110: (t1) :|[do]@5o3@m10@s2@h66@k-3p3v14
111: /*PART_A
112: (t1)
113: (t1) |:
114: (t1) |:2q7      b8<d16>a8b16g8a16f+8g16e8d8/*17 & 21     33 & 37
115: (t1)    q8      e8._2(e 16 <e)&e2.>~2      /*18 & 22     34 & 38
116: (t1)    q7      b8<d16>a8b16g8a16f+8g16e8d8/*19 & 23     35 & 39
117: (t1)    |1      e8d16>b2.&b16<          :| /*20          36
118: (t1) |2         e8d16>b2&b16b16<d16>b8:|   /*24          40
119: (t1) |: a8.<c8.e8.a4.&a16                  /*25 & 29     41 & 45
120: (t1)    g8f+16e16&e2.>                     /*26 & 30     42 & 46
121: (t1)    a8.<c8.e8.a4.&a16                  /*27 & 31     43 & 47
122: (t1)    g8f+16b16&b2.>          :|         /*28 & 32     44 & 48
123: (t1) @6~8:|
124: /*PART_B
125: (t1) @2o5q7~5
126: (t1) e8._3(e 16 <e)&e8.&e16&e2~3            /*50
127: (t1) d8.>b4&b16&b2                          /*51
128: (t1) e8._3(e 16 <c)&c8.&c16&c2~3            /*52
129: (t1) >b8.a4&a16&a2                          /*53
130: (t1) e8._3(e 16 <e)&e8.&e16&e2~3            /*54
131: (t1) d8.>b4&b16&b2                          /*55
132: (t1) e8._3(e 16 <c)&c8.&c16&c2~3            /*56
133: (t1) >b8.a4&a16&a2                          /*57
134: (t1) [loop]
135: /*-----------------------------------------
136: /*INTRO
137: (t2) l8o4@2v14@m3@s4                /*0
138: (t2) |: r4e2.d1                     /*1,2 & 9,10
139: (t2) r4e2.d2.dd                     /*3,4 & 11,12
140: (t2) >g1                            /*5   & 13
141: (t2) <d2.dd                         /*6   & 14
```

ンや音量バランス，モジュレーションなどの効果はあと回しで，とにかく楽譜本体をがしがし打ち込んでいく。

8時間後。なんとか目鼻がついてきたのでSEDIT.X[3]を立ち上げて音作りも並行して始める。SUPEREDとSEDIT，MMDSP.R[4]とMON.Rをいったりきたりしながら，徐々に曲は完成へと近づいていく。

さらに13時間後。やっと完成した。日本はすでに月曜日に突入。連続で作業していたのですっかり頭がくらくらしている。曲はイラストと違って，ちょっとでも油断すると思いついたフレーズが空気のように溶けていってしまい，もう二度と戻ってこないような気がして作業が中断できないのだ(これは私だけか?)。なんにせよ，完成したのはめでたい！　ってことで聴き直しは明日することにして今日は寝ることにする。お休みなさい。

X月XX日（月）

起きてパンとコーヒーを腹に入れたあと昨日の曲を聴き直す。案の定気に入らないところや，昨日は気づかなかった間違いがぼろぼろ見つかる。同じ曲をずーっと聴いてると麻痺しちゃうんだよなあ，としみじみ思いつつエディタで修正を開始。主にノイズとベースをいじる。モーツァルトにいわせれば「最初にひらめいたものにあとから手を加えるのは愚かな行為」なのだろうが，それは天才だからいえることなのであって，凡人の私は手を加えたくてたまらないのだ。たまらないんだったら。

さてすべての作業が完成した。結局できた曲が最初の目標であった「単調でかっこいい曲」とはまったくかけ離れた「ただ単に明るい曲」になっていることに気づく。ふん，人生そんなもんさ。とりあえず満足のいく曲が書けたのでOKを出す。

書いた曲を聴きながら

さて私の曲作りの様子を日記風に綴ってみましたが，いかがなものだったでしょうか？　だいたい私が曲を作るときは今回のように，わくわくして失敗してスランプになって，でもしばらく我慢して完成するというパターンが多いですね。これが毎回相当なエネルギーを使うわけで，完成したあとはたいてい消耗し切っているのですが，書いた曲を聴いてるとまたいろいろ思いついて，新しい曲を書きたくなっちゃうんですよね。なんにしても，ものを作る作業ってのはそういうもんだと思いますが。

```
142: (t2) >a1                              /*7   & 15
143: (t2) <d1                              /*8   & 16
144: (t2) :|[do]@5o3@m10@s2@h66@kp3v14
145: /*PART_A
146: (t2)
147: (t2) |:
148: (t2) |:2q7      b8<d16>a8b16g8a16f+8g16e8d8/*17 & 21     33 & 37
149: (t2)    q8      e8._2(e 16 <e)&e2.>~2     /*18 & 22     34 & 38
150: (t2)    q7      b8<d16>a8b16g8a16f+8g16e8d8/*19 & 23     35 & 39
151: (t2)    |1      e8d16>b2.&b16<           :| /*20          36
152: (t2) |2         e8d16>b2&b16b16<d16>b8:|    /*24          40
153: (t2) |: a8.<c8.e8.a4.&a16                  /*25 & 29     41 & 45
154: (t2)    g8f+16e16&e2.>                     /*26 & 30     42 & 46
155: (t2)    a8.<c8.e8.a4.&a16                  /*27 & 31     43 & 47
156: (t2)    g8f+16b16&b2.>            :|       /*28 & 32     44 & 48
157: (t2) @6~8:|
158: /*PART_B
159: (t2) @2o5q7~5
160: (t2) e8._3(e 16 <e)&e8.&e16&e2~3        /*49
161: (t2) d8.>b4&b16&b2                      /*50
162: (t2) e8._3(e 16 <c)&c8.&c16&c2~3        /*51
163: (t2) >b8.a4&a16&a2                      /*52
164: (t2) e8._3(e 16 <e)&e8.&e16&e2~3        /*53
165: (t2) d8.>b4&b16&b2                      /*54
166: (t2) e8._3(e 16 <c)&c8.&c16&c2~3        /*55
167: (t2) >b8.a4&a16&a2                      /*56
168: (t2) [loop]
169: /*==============================================
170: /*chord
171: (t3) l8o3@2v10                    /*0
172: (t3) |:~8@2a1&a2.&a8r8            /*1,2 & 9,10
173: (t3) a1&a1_8                      /*3,4 & 11,12
174: (t3) @1c1                         /*5   & 13
175: (t3) >b1                          /*6   & 14
176: (t3) a1                           /*7   & 15
177: (t3) d1<                          /*8   & 16
178: (t3) :|
179: /*PART_A
180: (t3) [do]
181: (T3) |:
182: (t3) v13@8p3o5q6@m0
183: (t3) |: r8b8a8b16<d16r16d16>b8a16r16b8  /*17 & 21        33 & 37
184: (t3)    r8b8a8b16<e16r16e16>b8a16r16b8  /*18 & 22        34 & 38
185: (t3)    r8b8a8b16<d16r16d16>b8a16r16b8  /*19 & 23        35 & 39
186: (t3)    r8b8a8b16<e16r16e16>b8a16r16b8:|/*20 & 24        36 & 40
187: (t3) q8o4v15@m7@s3@9
188: (t3) |: c1             /*25 & 29       41 & 45
189: (t3)    e2.g8f+8       /*26 & 30       42 & 46
190: (t3)    c1             /*27 & 31       43 & 47
191: (t3)    e2.b8a8 :|     /*28 & 32       44 & 48
192: (T3) :|
193: /*PART_B
194: (t3) q8o2v13@m4@s2@10p2
195: (t3) e1                /*49
196: (t3) g1                /*50
197: (t3) c1                /*51
198: (t3) d1                /*52
199: (t3) e1                /*53
200: (t3) g1                /*54
201: (t3) c1                /*55
202: (t3) d1                /*56
203: (t3) [loop]
204: /*----------------------------------------------
205: (t4) l8o4@2v10                    /*0
206: (t4) |:~8r8@2d2&d4.&d2.&d8r8      /*1,2 & 9,10
207: (t4) r8d2&d4.&d1_8                /*3,4 & 11,12
208: (t4) >@1@k8c1                     /*5   & 13
209: (t4) >b1                          /*6   & 14
210: (t4) a1                           /*7   & 15
211: (t4) d1@k0<<                      /*8   & 16
212: (t4) :|
213: /*PART_A
214: (t4) [do]
215: (t4) |:
216: (t4) v14@8p3o5q6@m0
217: (t4) |: r8g8g8g16b16r16b16g8g16r16g8    /*17 & 21        33 & 37
218: (t4)    r8g8g8g16b16r16b16g8g16r16g8    /*18 & 22        34 & 38
219: (t4)    r8g8g8g16b16r16b16g8g16r16g8    /*19 & 23        35 & 39
220: (t4)    r8g8g8g16b16r16b16g8g16r16g8:|  /*20 & 24        36 & 40
221: (t4) q8o3v16@m7@s3@9
222: (t4) |: a1             /*25 & 29       41 & 45
223: (t4)    b1             /*26 & 30       42 & 46
224: (t4)    a1             /*27 & 31       43 & 47
225: (t4)    b1       :|    /*28 & 32       44 & 48
226: (t4) :|
227: /*part_b
228: (t4) q8o2v13@m4@s2@10@k-5p1
229: (t4) e1                /*49
230: (t4) g1                /*50
231: (t4) c1                /*51
232: (t4) d1                /*52
233: (t4) e1                /*53
234: (t4) g1                /*54
235: (t4) c1                /*55
236: (t4) d1                /*56
237: (t4) [loop]
238:
239: /*==============================================
240: /*melody_sub
241: /*INTRO
242: (t5) @k318o4@2r*11p2v13@m3@s6    /*0             r*11:echo
243: (t5) |: r4g2.f+1                  /*1,2 & 9,10
244: (t5) r4g2&g8a16g16f+2.f+f+        /*3,4 & 11,12
245: (t5) e2&e8ede                     /*5   & 13
246: (t5) f+2.f+f+                     /*6   & 14
247: (t5) e2&e8ede                     /*7   & 15
248: (t5) g2f+2                        /*8   & 16
249: (t5) :|[do]@5o3@m15@s2@h66@k0p2@v114
250: /*PART_A
251: (t5)
252: (t5) |:
253: (t5) |:2q7      b8<d16>a8b16g8a16f+8g16e8d8/*17 & 21     33 & 37
```

おりしも季節は夏真っ盛り。本文中で紹介した素晴らしいツールたちも待っていることですし，一発作曲でも始めてみませんか？　最初は出来なんてどうでもいいですから，自分だけの世界を音に映してみましょうよ。きっと68君は新しい世界を見せてくれると思いますよ。

1)　RCD.XはHARPOON氏の手によるMIDI音源演奏ドライバ（現在はTURBO氏がバージョンアップを引き継いでいる）。RCM-PC98のRCPフォーマットデータをそのままで聴くことができるので大変便利。また各種コンバータも充実しており，たいていのフォーマットのMIDIデータが利用できる素晴らしい環境が構築されている。またZ-MUSICにもNOVA氏制作のコンバータZtoR.Xが用意されている（ZMDファイルからRCPファイルを生成）。私はRCD ver.2.92jを使用させてもらっている。

STED2.XはそのRCD.X上で動作するステップ入力中心のミュージックエディタ（リアルタイム録音も可能）。こちらはTURBO氏制作で，私はver.2.02を愛用させてもらっている。入力中の楽譜表示や他トラックを参照しながらの入力もできて，私はMUSICPRO-68K［MIDI］からすっかりこちらに乗り換えてしまった。とても快適。

RCD.X，STED2.Xはどちらもパソ通上で入手可能なフリーウェア。これだけ高機能なソフトがユーザーの手によって提供されているというのも，X68000ならではのありがたい環境だろう（感謝感謝）。

2)　SUPERED.XはED.Xをベースに大幅な改良，高速化を施したフルスクリーンエディタ。作者はT.Nishikawa氏。外字定義によりPC-9801文字を表示できたりもするので通信にも重宝している。Z-MUSICで曲を書くときには西川氏の作った便利なマクロも用意されており，これを利用するとSUPEREDから抜けなくても編集中の曲を演奏することができるので大変便利である（以前は行頭に移動してESC＋W OPM［RET］を自分でやらなければならなかったので面倒だったのだ。またファイルネームをOPMにしたままESC＋Eをしてしまい，何回かファイルを失ったことがある）。

3)　SEDIT.Xはおーちゃん氏作のFM音色エディタ。名前が似ているがSOUND PRO 68Kとは無関係。SNDファイルの読み込みのみならず，テキスト形式のファイルを直接読み込んでエディットすることが可能になっている。また音色を作りながら，キーボードのキーボード（鍵盤）で和音を出せるようになっているので音の確認も簡単だ。さらにエフェクト（デチューン，エコー）の設定もできるようになっているので，曲中で使用したときのイメージもしっかりつかめるように工夫されている。私はver.2.3にいつもお世話になってる。SEDIT.Xもパソ通上で手に入るフリーウェア。

4)　MMDSP.Rは複数音源ドライバに対応したリアルタイムディスプレイ。制作はMiahmie氏とGao氏とかるちゃん氏。美しいスペクトラムアナライザや端正なレイアウト，色づかいなどで，数あるディスプレイの中でも群を抜いた美しさを誇っている。

演奏機能も充実しており，プログラム演奏，シャッフル演奏，オート演奏，イントロ演奏など考えられるものはすべて入っている。もちろんスキップ，ポーズ，マスクも思いのまま。

対応しているドライバはMXDRVにMADRV，MLDにRCD，そしてZ-MUSICとX68000上のあらかたのドライバをサポート（まだいっぱいあるといえばあるんだけどね）。早くZ-MUSICにも完全対応してほしい。

```
254: (t5)    q8      e8._2(e 16 <e)&e2.>~2           /*18 & 22    34 & 38
255: (t5)    q7      b8<d16>a8b16g8a16f+8g16e8d8/*19 & 23    35 & 39
256: (t5)    |1      e8d16>b2.&b16<           :| /*20          36
257: (t5) |2         e8d16>b2&b16b16<d16>b8:|   /*24          40
258: (t5) |: a8.<c8.e8.a4.&a16                   /*25 & 29    41 & 45
259: (t5)    g8f+16e16&e2.>                      /*26 & 30    42 & 46
260: (t5)    a8.<c8.e8.a4.&a16                   /*27 & 31    43 & 47
261: (t5)    g8f+16b16&b2.>              :|      /*28 & 32    44 & 48
262: (t5) @6p2~8:|
263: /*PART_B
264: (t5) @2o5q7p2~5
265: (t5) e8._3(e 16 <e)&e8.&e16&e2~3          /*50
266: (t5) d8.>b4&b16&b2                         /*51
267: (t5) e8._3(e 16 <c)&c8.&c16&c2~3          /*52
268: (t5) >b8.a4&a16&a2                         /*53
269: (t5) e8._3(e 16 <e)&e8.&e16&e2~3          /*54
270: (t5) d8.>b4&b16&b2                         /*55
271: (t5) e8._3(e 16 <c)&c8.&c16&c2~3          /*56
272: (t5) >b8.a4&a16&a2                         /*57
273: (t5) [loop]
274: /*============================================
275: /*装飾
276: (t6) 18o0@3v14p1q7       /*0
277: (t6) |:8r1:|                          /*1-8
278: (t6) d1&d2.a4                         /*9,10
279: (t6) d1&d1<                           /*11,12
280: (t6) c4<c4>b4e4                       /*13
281: (t6) >b4b4<d4f+4                              /*14
282: (t6) e4e4e4e4                         /*15
283: (t6) d4d4d4d4                         /*16
284: /*PART_A (こっからベース)
285: (t6) [do]
286: (t6) 116@7o2@v127p3q6
287: (t6) |:
288: (t6) |: ggggg32g32gggggggg<d>gb      /*17 & 21     33 & 34 G
289: (t6)    eeeee32e32eeeeeeee<e>eb      /*18 & 22     35 & 36 Em
290: (t6)    ggggg32g32gggggggg<d>gb      /*19 & 23     37 & 38 G
291: (t6)    eeeee32e32eeeeeeee<e>eb:|    /*20 & 24     39 & 40 Fm
292: (t6) |: aaaaa32a32aaaaaaaaaa         /*25 & 29     41 & 42 Am
293: (t6)    eeeee32e32eeeeeeee<e>d<d>    /*26 & 30     43 & 44 Em
294: (t6)    aaaaa32a32aaaaaaaaaa         /*27 & 31     45 & 46 Am
295: (t6)    eeeee32e32eeeeeeee<e>d<d>:|/*28 & 32     47 & 48 Em
296: (t6) :|
297: /*PART_B
298: (t6) r1          /*49
299: (t6) r1          /*50
300: (t6) r1          /*51
301: (t6) r1          /*52
302: (t6) r1          /*53
303: (t6) r1          /*54
304: (t6) 132         <c8cccrcrcrcrc8cccrcrcrcr             /*55
305: (t6)             d8dddrdrdrdrdr
306: (t6)             d16<d16>e16<e16>f+16<f+16>a16<a16>    /*56
307: (t6) [loop]
308: /*--------------------------------------------
309: (t7) 18o0@4v14p2q7                    /*0
310: (t7) |:8r1:|                          /*1-8
311: (t7) d1&d2.a4                         /*9,10
312: (t7) d1&d1<                           /*11,12
313: (t7) c4<c4>b4e4                       /*13
314: (t7) >b4b4<d4d4                       /*14
315: (t7) e4e4e4e4                         /*15
316: (t7) d4d4d4d4                         /*16
317: /*PART_A (こっからchordの手伝い=t3のデチューン)
318: (t7) [do]
319: (t7) |:
320: (t7) v13@8p3o5q6@k7p1@m0
321: (t7) |: r8b8a8b16<d16r16d16>b8a16r16b8  /*17 & 21     33 & 37
322: (t7)    r8b8a8b16<e16r16e16>b8a16r16b8  /*18 & 22     34 & 38
323: (t7)    r8b8a8b16<d16r16d16>b8a16r16b8  /*19 & 23     35 & 39
324: (t7)    r8b8a8b16<e16r16e16>b8a16r16b8:|/*20 & 24     36 & 40
325: (t7) q8o4v15@m7@s3@9@K3p1
326: (t7) |: c1             /*25 & 29     41 & 45
327: (t7)    e2.g8f+8       /*26 & 30     42 & 46
328: (t7)    c1             /*27 & 31     43 & 47
329: (t7)    e2.b8a8 :|     /*28 & 32     44 & 48
330: (t7) :|
331: /*PART_B
332: (t7) q8o2v10@m4@s2@10p3r8                /*r8=echo
333: (t7) e1                /*49
334: (t7) g1                /*50
335: (t7) c1                /*51
336: (t7) d1                /*52
337: (t7) e1                /*53
338: (t7) g1                /*54
339: (t7) c1                /*55
340: (t7) d2.&d8            /*56                /*echo解除
341: (t7) [loop]
342: /*============================================
343: /*ドラムその他
344: (t8) @11v14o3
345: (t8) |:8r1:|                      /*1-8
346: (t8) p1c1&c2r2p2c1&c2r2           /*9-12
347: (t8) p1c1&c2r2p2c1&c2r2           /*13-16
348: (t8) [DO]@v127116
349: /*PART_A
350: (t8) |: 63o2_3@12g8~3@13d_3@12brb~3@13d8:|      /*17-47
351: (t8) o4@v127@14s0@s10@m-150p2
352: (t8) g32g32g32g32rgp3e32e32e32e32p1c32c32c32c32@m0 /*48
353: /*PART_B
354: (t8) @11v13o3c1&c2r2     /*49,50
355: (t8) c1&c2r2     /*51,52
356: (t8) c1&c2r2     /*53,54
357: (t8) o2@12_3     g8g8g8g8g8g8g8g8                    /*55
358: (t8)             g8g8g8g8
359: (t8)             o4@v127@14s0@s10@m-150p1
360: (t8)         g32g32g32g32rgp3e32e32e32e32p2c32c32c32c32@m0 /*56
361: (t8) [loop]
362:
363: (p)
364: .comment NEW EIGHT -新しい夜明けに-  COMPOSED BY T.Takahashi
365: .comment                              93.06.15 TUE 06:18
```

タッチタイピングへの野望

ダラダラいこう

Itou Masahiko 伊藤 雅彦

思い立ったらプログラミング。ぼんやりアルゴリズムを考え，行き詰まったらテレビでも見ながらリラックス。本来の目的を忘れてテレビに熱中してもいいじゃないか。時間を気にせずダラダラいこうよ。

タッチタイピングしたい

へっ，どうせ私はタッチタイピングができませんよ。

パソコンに取り憑かれてはや10年。QWERTY配列のキーボードには，ずいぶんとお世話になってきたわけだけど，いまだにタッチタイピングはできない。ま，ぐうたらな私だから，パソコンにそれほど心血注いできたわけでもないし，キーボードを見ながらでもそこそこ速く打てたからそれで十分だったんだ。「Wのキーはどこだっけ」なんて，キーボードの上を指先がウロウロすることはないしね。打ちたいキーにすぐ目線がいく。目でキー配列を覚えている感じかな。

たまに会社で，おっさんが明らかに慣れていない手つきでワープロを打っているのを見かける。ちょくちょく手の動きを止めて，キーボードのあちこちに目線を泳がせているのを見ると，なんだかおかしさが込み上げてきちゃうな。しかし，私もパソコンを始めたてのころは，あんな滑稽な姿でキーボードと格闘して「Cはどこだ C，C！」と心の中で叫んでいたっけ。

でも，目でキー配列を覚えた現在でも，それだけではタッチタイピングできないみたい。そりゃそうか，見ないで打つのがタッチタイピングだから，目で覚えたってだめだろうね。

しかし，見ないで打とうと思ってもなまじ目で覚えているから，ついつい目線がキーボードにいっちゃう。目じゃなくて指で覚えるものだもんな。結局は，ただキーボードを触っているとできるようになるものじゃなくて，それなりに訓練しなくちゃならないのは当然か。

ローマ字入力だけでも

そういったわけでタイピング教本を1冊買ってくる(参考文献)。英文タイプのものではなくて，日本語ローマ字入力の教習本にした。どっちでも同じようなものだと思っていたらそうでもなくて，英文タイプができればローマ字入力もできるけど，ローマ字入力ができても英文タイプができるわけではない，と書いてある。それから，ローマ字入力のほうがはるかに習得しやすいと書いてある。

それなら，とりあえずローマ字入力を練習しよう。英文タイプなんてどうだっていいや。プログラムのコーディングなんて，だらだらやるんだからね。それよりもワープロなんかで文章をシュパパパッと華麗に打てたほうが幸せなような気がする。

ということで，ここはひとつローマ字タイピング練習プログラムでも作ってみますか。自作プログラムを使ってなら，少しは根気よく練習できるかもしれないし。

電源を入れて考える

X1turboZIIの電源を入れて，turbo BASICを起動する。このご時世にX1のBASICでプログラムを組むのは，別に読者のみなさんに嫌がらせをしたいのではなくて，私がX68000を持っていないから，という単純な理由。

ま，X1特有の技を駆使するわけでもなし，X-BASICがわかる人ならこのプログラムリストだって読めると思う。気が向けば移植だって簡単。

さて，タイピング練習プログラムというものは，練習用の短文を画面に表示してその短文をローマ字入力させ，入力が間違っていたらピッと警告音を出して再入力させる。そして，最後まで正しく入力されたら，「よかったね，さあもう一度練習しましょう」てな具合で，また短文を入力させることを繰り返していけばいいわけだ。

簡単そうだけど，これだけの考えでは大まかすぎてプログラムは1行も書けない。せっかく立ち上げたBASICをほったらかしにして，しばし考える。

ローマ字入力ってものは，アルファベットを入力していくと音ごとにカナに変換されていく。皆さんご承知のとおりだ。

それなら，配列を用意してカナとローマ字を音ごとに格納しておこう。たとえば，

```
KANA$(0)="と",ROMA$(0)="TO"
KANA$(1)="う",ROMA$(1)="U"
KANA$(2)="きょ",ROMA$(2)="KYO"
KANA$(3)="う",ROMA$(3)="U"
```

という具合にね。こうしておけば，あとあとの処理が楽そうな気がする。でも練習用の短文をDATA文で用意するときに，いちいち，

```
DATA と，う，きょ，う
DATA TO,U, KYO,U
```

なんていうふうに書いていたら面倒臭い。あくまでDATA文は，

```
DATA とうきょう
```

という具合にスッキリと書くべきだ。そうやって用意した短文を音節ごとに区切って，配列に格納してやりたいな。よーし，まずは短文の文字列を音節に区切って，さらにその音節をローマ字に変換するルーチンを作ってみよう。

INSTR関数がおいしい

カナ文字はたいてい1文字で1音を表しているけど，「きゃ」「じょ」みたいに拗音が入ると2文字で1音を表すことになる。これを念頭に置きつつ，まずはカナ，ローマ字対応表を用意する(リスト1，1060〜1470行)。

ここで，速度稼ぎの工夫をしてみた。注目してほしいのはKANA1$とKANA2$。1文字で1音のカナはKANA1$に“あいうえおかき……”という形で入り，2文字で

1音のカナは，KANA2$に“きゃきゅきょしゃ……”となるように格納している。

こうしておいて，あとでカナ，ローマ字対応表を参照するときに，INSTR関数を使う。この関数は文字列の中から特定の文字列パターンを探して，何文字目にそのパターンがあったか返してくれるもの。つまり，文字列検索をやってくれる関数なわけで，これを使えばBASICで検索ルーチンを組むよりも数段速い検索ができちゃう。BASICを使っていると速度で不満が出ることが多いから，この関数はおいしく使えるかもしれない。ひとついいものを見つけてちょっと嬉しい気分。

カナ，ローマ字対応表ができたところで，本題のルーチンである短文を音に区切って配列に入れていくルーチンを作ってみよう。元の短文はBUN$に入っていることにして，それを音に区切ったものをBUNKANA$(N)に入れ，さらにそれに対応するローマ字をBUNROMA$(N)に入れるのだ（リスト2，1800～2150行）。

まず，BUN$の頭の2文字を切り出してきて，その2文字がカナ，ローマ字対応表のカナ欄にあるかどうか調べる。さっき見つけたINSTR関数のおいしい使い方で，

INSTR(1,KANA2$,切り出した2文字)

とやるのだ。もし，検索が成功すれば切り出した2文字が2文字で1音を表すようなカナ文字だ，ということになる。検索に失敗したら，今度は頭の1文字だけを切り出して，同じようにKANA1$を検索する。普通は，ここで必ず検索に成功するはずだから，もしも成功しなかったときには「BUN$に入っている短文が変だよ」とエラーを出してしまおう。

こんな具合に処理を短文の最後の文字まで繰り返していけば，短文を音に区切れる。ローマ字変換だってINSTR関数の返り値を使って，ローマ字の入っている配列を参照すればいいわけだ。

これだけ考え方がまとまればプログラムを入力できる。本当はプログラムを入力しながら考えをまとめていたんだけど，仕事でプログラムを組んでいるわけじゃないからいいじゃないか。入力しては消し，また入力しては消し，なんてことをやりながらアルゴリズムを考えるのも楽しいものだ。

そんなわけで，考えがまとまったときにはプログラムもできていた。よかったよかった。さて，次の処理はなにかな～っと。

促音の処理

と，順調に次へ進みたいところだけど，まだひとつ問題が残っているんだな。先ほどいったやり方では促音に対応できないんだ。

最初からこの問題に気づいていなかったわけじゃないけど，これは例外的なことだ，と割り切ってとりあえず横に置いておいた。要するになにかしらのプログラムができあがってから，促音用の修正を加えようと思っていたんだ。最初から例外処理のことを考えていると，頭がこんがらがっちゃう場合があるからね。もっとも，あとで例外処理の修正ができないようなプログラムを組んじゃって，最初から作り直しってこともあるけど，そのときには必ず代案が思い浮かぶだろうから，最初から作り直すのもそんなに大変じゃない。

なんていっているうちに，なんとかうまい促音処理の修正方法を思いついた。促音を入力するときには，そのあとに続く文字のローマ字の最初のアルファベットを連打するわけでしょ。それなら，促音はその直後の音と一緒にしてひとつの音として扱おう。たとえば「えっくす」なら，

え　っく　す
E　KKU　SU

と区切る。「らっきょう」なら，

ら　っきょ　う
RA　KKYO　U

という具合だ。そんなこんなで，プログラムをちょこちょこっと修正してできあがり。細かいところを突っつけば，ちょっと問題もあるんだけど大丈夫でしょう。

よし，これでできた。この部分ができてしまえば，あとの処理は屁みたいなもの。アルゴリズムなんか考えなくても，いきあたりばったりでプログラムが組めそうだ。

とりあえず動いた

以上で作ったルーチンはいわば下ごしらえの部分。今度はメイン処理，短文をローマ字入力させるルーチンを作るぞ（リスト3，1750～1790，2180～2470行）。

まずは画面構成を考えよっと。プログラムを作っていて，どういう画面にしようかって考えているときはいつも楽しい。今回は，どんなふうにブラウン管を光らせようかな。

と思ったけど，今回はものがものだけに

▼リスト1

```
1000 ' タ イ ピ ン グ 練 習
1010 '
1020 WIDTH 80,25,0,2:INIT:DEFINT A-Z
1030 '
1040 DIM ROMA1$(74), ROMA2$(46), BUNROMA$(126), BUNKANA$(126), BUNX(126)
1050 DIM REIBUN$(9), BUNMISS(126)
1060 '
1070 KANA1$=""
1080 FOR I=0 TO 74
1090   READ I$,ROMA1$(I)
1100   KANA1$=KANA1$+I$
1110 NEXT
1120 DATA あ, A,い, I,う, U,え, E,お, O
1130 DATA か, KA,き, KI,く, KU,け, KE,こ, KO
1140 DATA さ, SA,し, SI,す, SU,せ, SE,そ, SO
1150 DATA た, TA,ち, TI,つ, TU,て, TE,と, TO
1160 DATA な, NA,に, NI,ぬ, NU,ね, NE,の, NO
1170 DATA は, HA,ひ, HI,ふ, FU,へ, HE,ほ, HO
1180 DATA ま, MA,み, MI,む, MU,め, ME,も, MO
1190 DATA や, YA,       ゆ, YU,       よ, YO
1200 DATA ら, RA,り, RI,る, RU,れ, RE,ろ, RO
1210 DATA わ, WA,                     を, WO
1220 DATA が, GA,ぎ, GI,ぐ, GU,げ, GE,ご, GO
1230 DATA ざ, ZA,じ, JI,ず, ZU,ぜ, ZE,ぞ, ZO
1240 DATA だ, DA,ぢ, DI,づ, DU,で, DE,ど, DO
1250 DATA ば, BA,び, BI,ぶ, BU,べ, BE,ぼ, BO
1260 DATA ぱ, PA,ぴ, PI,ぷ, PU,ぺ, PE,ぽ, PO
1270 DATA ヴ, VU,ん, X,ー, ¥,、,",",。, .
1280 '
1290 KANA2$=""
1300 FOR I=0 TO 46
1310   READ I$,ROMA2$(I)
1320   KANA2$=KANA2$+I$
1330 NEXT
1340 DATA きゃ,KYA,              きゅ,KYU,              きょ,KYO
1350 DATA しゃ,SYA,              しゅ,SYU,しぇ,SHE,しょ,SYO
1360 DATA ちゃ,TYA,              ちゅ,TYU,ちぇ,CHE,ちょ,TYO
1370 DATA にゃ,NYA,              にゅ,NYU,              にょ,NYO
1380 DATA ひゃ,HYA,              ひゅ,HYU,              ひょ,HYO
1390 DATA みゃ,MYA,              みゅ,MYU,              みょ,MYO
1400 DATA りゃ,RYA,              りゅ,RYU,              りょ,RYO
1410 DATA ぎゃ,GYA,              ぎゅ,GYU,              ぎょ,GYO
1420 DATA じゃ, JA,              じゅ, JU,じぇ, JE,じょ, JO
1430 DATA ぢゃ,DYA,              ぢゅ,DYU,              ぢょ,DYO
1440 DATA びゃ,BYA,              びゅ,BYU,              びょ,BYO
1450 DATA ぴゃ,PYA,              ぴゅ,PYU,              ぴょ,PYO
1460 DATA ふぁ, FA,ふぃ, FI,              ふぇ, FE,ふぉ, FO
1470 DATA ヴァ, VA,ヴィ, VI,              ヴェ, VE,ヴォ, VO
```

表示するものがあまりない。画面中央あたりに短文を表示して，その短文の２行下ぐらいでカーソルを点滅させてキー入力を待つようにする。キーが押されたら押したアルファベットを表示し，１音節分のローマ字が入力されたところでアルファベットがカナ文字にパッと変わる。そうやって，日本語FEPでローマ字カナ変換をするように画面に短文を書いていく。短文を最後まで打ち込んだとき，画面中央には同じ短文が上下２つ並んで表示されることになる。

これだけでいいか。ほかに表示するものもないしね。あとでちょっと表示したいものがあるけど，それ以外は小細工しないでこのシンプルな画面でいこう。ま，その代わりといっちゃあなんだけど，表示される短文はちゃんと真ん中に出てくるようにしてあげよう。

で，これぐらいイメージを固めて，再びプログラムの入力に取りかかる。簡単な文字列比較や文字列表示ばっかりだ。う～む，それぞれの音が何桁目に表示されているかがわかっていると便利そうだな。BUNX$(N)という配列を用意して，そこに各音のX桁目を入れておこう。さっき作った下ごしらえのルーチンに手を加えちゃえば簡単だ。

フフフン，とできた。これでとりあえず実行できるだけのルーチンが揃ったな。よし，実行してみよう。おっ，カーソルが変なところに出てきてる。そのほかは大丈夫かな。大丈夫そうだから，さっそくデバッグだ。デバッグってのはなんだかんだいっても楽しいもんだな。いろんなバグ君に出合えるし。特に私は，画面が華やかに崩れていくバグが好きだ。思いがけない画面を目の当たりにして，思わず「おおっ！」と叫んでしまうあの瞬間がたまらない。

でも，今回のプログラムは単純だからそれほどたいしたバグが出るわけもなく，小さなバグ君たちとひとしきり遊んだだけで終わってしまった。ちょっと残念。

飾りつけ

さあて，肝心の部分は完成したから，今度は飾りをドンドンつけていこう。

まず，プログラムを走らせたら何種類かの短文をメニュー表示して，その中から練習する短文を選べるようにしないといけないな（リスト４，1480～1740，2720～2730行）。選べる短文は10種類にして，テンキーで選ぶようにしよう。DATA文で短文を用意して，っと10個も短文を考えなきゃならないのか。ま，どんな文だっていいんだけど，ついつい時間をかけてあれこれ考えてしまう。

それから，短文の入力が終わったら入力にかかった時間と，入力ミスした回数を表示する機能も欠かせない。ついでに入力ミスした場所も表示してしまおう（リスト５，2160～2170，2480～2710行）。それから，時間表示にはハイスコア処理も加えて……タイピング練習プログラム，見事完成。

使い方

ここで，このプログラムの入力方法と改造方法を説明しよう。

入力方法は，turbo BASIC上で打ち込む，それだけ。説明の関係上，かなりリストがバラバラになっているので，行番号に注意しながら打ち込んでもらいたい。

で，1120～1270行と1340～1470行のカナ，

▼リスト２

```
1800 '
1810 BUN$=BUN$+"@"
1820 MOJIME=1
1830 SOKUON=0
1840 UNITNO=0
1850 WHILE MID$(BUN$,MOJIME*2-1,2)<>"@"
1860   I=INSTR(1,KANA2$,MID$(BUN$,MOJIME*2-1+SOKUON*2,4))
1870   IF I<>0 ELSE 1920
1880     I$=ROMA2$((I-1)/4)
1890     KANALEN=2
1900     I=1
1910     GOTO 2050
1920   'ELSE
1930     I$=MID$(BUN$,MOJIME*2-1+SOKUON*2,2)
1940     I=INSTR(1,KANA1$,I$)
1950     IF I<>0 ELSE 2000
1960       I$=ROMA1$((I-1)/2)
1970       KANALEN=1
1980       I=1
1990       GOTO 2040
2000     'ELSE
2010       IF I$<>"っ" THEN PRINT:PRINT "文が変！":BEEP:END
2020       SOKUON=SOKUON+1
2030       I=0
2040     'ENDIF
2050   'ENDIF
2060   IF I=1 ELSE 2130
2070     BUNROMA$(UNITNO)=STRING$(SOKUON,LEFT$(I$,1))+I$
2080     BUNKANA$(UNITNO)=MID$(BUN$,MOJIME*2-1,(KANALEN+SOKUON)*2)
2090     BUNX(UNITNO)=BUNTOPX+MOJIME*2-2
2100     MOJIME=MOJIME+KANALEN+SOKUON
2110     UNITNO=UNITNO+1
2120     SOKUON=0
2130   'ENDIF
2140 WEND
2150 UNIT=UNITNO
```

▼リスト３

```
1750 BUNTOPX=(80-LEN(BUN$))/2
1760 BESTTIME=999
1770 CLS
1780 LOCATE BUNTOPX,8
1790 PRINT BUN$
2180 '
2190 FOR I=0 TO UNIT-1
2200   BUNMISS(I)=0
2210 NEXT
2220 MISS=0
2230 UNITNO=0
2240 TIME=0
2250 WHILE UNITNO<UNIT
2260   INKY$=""
2270   REPEAT
2280     LOCATE BUNX(UNITNO),10
2290     COLOR 6
2300     PRINT INKY$;
2310     COLOR 7
2320     INKY$=INKY$+INKEY$(1)
2330     IF LEFT$(BUNROMA$(UNITNO),LEN(INKY$))<>INKY$ ELSE 2400
2340       BEEP
2350       LOCATE BUNX(UNITNO),10
2360       PRINT SPACE$(LEN(INKY$)-1)
2370       BUNMISS(UNITNO)=1
2380       MISS=MISS+1
2390       INKY$=""
2400     'ENDIF
2410   UNTIL BUNROMA$(UNITNO)=INKY$
2420   LOCATE BUNX(UNITNO),10
2430   COLOR 5
2440   PRINT BUNKANA$(UNITNO)+" "
2450   COLOR 7
2460   UNITNO=UNITNO+1
2470 WEND
```

ローマ字対応表のローマ字データは，各自の好みに合わせて修正するといいだろう。ローマ字には，訓令式とヘボン式の2つの入力方法があるので，自分の慣れているほうで練習しないと意味がないだろうからね。とりあえず，リストでは訓令式でデータを作ってある。

ま，こういう仕様は日本語FEPなら完全に失格，論外なんだけど，タイピング練習プログラムとしては，むしろどちらかの方法で統一したほうがよいと考えたからだ。同じ文字を入力するのにいつも違う打鍵をしていたら，タイピング速度はなかなか速くならないだろうからね。

カナ，ローマ字対応表を自分流に合わせたら，今度は1520行からの短文データを適当に変更しておくといいかもね。掲載リストにあるデータは，私がふざけ半分に考えたものだからいまいち実用性はないかも。ここはやっぱり，タイピング教習本を参考にして，自分の実力に合った練習効果の高いものを考えるといいだろう。短文の最大文字数は40文字となっている。

ただし，1610行のデータは，私が買った教習本からちょっと拝借したものだ。著者自ら「これ以上，練習に適したものはありません」と断言しているほどのものなので，これを使って練習を積むと効果がそれなりに表れるかもしれない。

というように，準備が終わったらあとは実行するだけ。実行するときにはCAPSロックキーをON，カナロックキーをOFFにしておくように。なお，プログラムを終了したいときには，素直にSHIFT＋BREAKで実行を止めていただきたい。

ちなみに，このプログラムはカナ，ローマ字対応表のデータを変えることで，かな入力の練習にも使える。やり方は，対応表のローマ字を半角カタカナに変えるだけだ。

▼リスト4

```
1480 '
1490 FOR I=0 TO 9
1500   READ REIBUN$(I)
1510 NEXT
1520 DATA ろーまじたいぴんぐのれんしゅうをしましょう。
1530 DATA うつくしいひとにめぐりあいたい、うつくしいひとにみつめられたい。
1540 DATA ろだんのかんがえるひと、はためくひのまる、まわるちきゅうぎ
1550 DATA ひしょうきゃくだ、あたたたた、ああきぶんそうかい。
1560 DATA おおぞらをかけまわるいつつぼし、それはあおいしょうげき
1570 DATA あさっぱらからとてもようきな、はやみゆうのあめりかんきっず
1580 DATA こんばんは、たわらこうたろうです。
1590 DATA ええっめがどらいぶでヴァーちゃれーしんぐができるの
1600 DATA あいうえおあいうえおいえあおういえあおう
1610 DATA さいこうのせいしょうに、することでした。
1620 '
1630 WHILE 0=0
1640 '
1650 CLS
1660 FOR I=0 TO 9
1670   PRINT I;":";REIBUN$(I)
1680   PRINT
1690 NEXT
1700 PRINT "数字キーで文を選択してください";
1710 REPEAT
1720   I=ASC(INKEY$(1))-&H30
1730 UNTIL I>=0 AND I<=9
1740 BUN$=REIBUN$(I)
2720 '
2730 WEND
```

▼リスト5

```
2160 '
2170 REPEAT
2480 '
2490 RECORD=TIME
2500 COLOR 2
2510 FOR I=0 TO UNIT-1
2520   IF BUNMISS(I)=1 ELSE 2550
2530     LOCATE BUNX(I),10
2540     PRINT BUNKANA$(I)
2550   'ENDIF
2560 NEXT
2570 COLOR 7
2580 IF RECORD<BESTTIME THEN BESTTIME=RECORD
2590 LOCATE 60,6
2600 PRINT "最高記録：";BESTTIME;"秒  "
2610 LOCATE 20,13
2620 PRINT "只今の記録、";RECORD;"秒で失敗回数は";MISS;"でした。";
2630 REPEAT
2640   I$=INKEY$(1)
2650 UNTIL I$=" " OR I$=CHR$(13)
2660 LOCATE 0,10
2670 PRINT SPACE$(80)
2680 LOCATE 0,13
2690 PRINT SPACE$(80)
2700 '
2710 UNTIL I$=CHR$(13)
```

ダラダラしようよ

今回，お気楽にプログラミングしようか，ということでダラダラとプログラムを作ってみた。内容的にも難しいことはなにもないので，プログラミングも楽々終わってしまった。しかし，ダラダラとプログラミングするのも難しいものだ。うっかりするとプログラミングのことを忘れて，ただ単にダラダラしてしまうだけになるからだ。

特に，TVチューナつきのディスプレイを使っていると，ついつい画面をテレビに切り替えちゃうからさらにいけない。ちょっとだけのつもりでも，Jリーグの試合中継なんかやっているとだんだん引き込まれちゃう。サッカーなんか見ていたってたいして面白くもないだろう，と思っていたが実際そうでもない。ついつい試合展開が気になってしまう。

んで，ぼけーっと観戦しているうちに高校2年生のときだったかな。体育の授業のサッカーのときには，いつもキーパーをやらされていたのを思い出した。試合時間のほとんどをゴール前にひとりたたずんで，浪費していくキーパーってなんなんだ。それで，たまにボールがきたときにミスをすると，それがそのまま失点に結びつくからチームメイトの視線が痛い。しょうがないだろ，さっきから走り回っているおまえらと違って，こっちは準備運動ができていないんだ。それになんで体育のサッカーは冬の真っ只中にやるんだ。からっ風がぴゅーぴゅー吹くグラウンドに突っ立って，鼻をすすってなきゃいけないキーパーのむなしさを考えたことがあるのか，先生さんよ。

なんてことを考えていると，プログラミングのことはすっかり忘れちゃう。やっぱり，ダラダラプログラミングするといっても，テレビを見ていても頭のどこかではアルゴリズムを煮詰めていて，それが形になったときに頭を切り替えてプログラム作成に戻っていく，そんな姿が理想なんだろうな。この姿こそ，プログラミングを楽しんでいる人の姿だよ，きっと。これができれば「趣味はプログラミングです」と胸を張っていえるんじゃないかな(本当か？)。さあ，君もダラダラプログラミングをしてみようじゃないか。

＜参考文献＞
「ブラインドタッチの達人」明石誠，日本文芸社

朝日が眩しいコンピュータライフ

通信中毒者から愛のメッセージ

Ishibumi Akira 伊溢見 あきら

コミュニケーション手段のひとつとして，パソコン通信が含まれるネットワーク社会には，独特の世界観，ルールが存在します。その通信の世界を伊溢見氏とともに，ちょっと垣間見てみましょう。

■

人間は情報によって文明を発展させてきました。知識を伝達することで情報とし，それを交互にやりとりすることで互いの知識を高めてきました。このことは，現在までの人類の発展すべてに共通していることです。そのための手段は，言葉や文字にその起源をもち，さらにそれをより遠く速く大量に伝達することを求めた結果，交通機関や電信電話，通信衛星という，文明の利器が次々と発明されて今日にいたっているのです。

■

最も新しいメディア

■

こういった，情報と人間の密接な関係をもっともらしく説明したあとで，情報時代の現在にその象徴であるコンピュータを使って情報伝達をやっている，つまりパソコン通信をしているといえば，言葉だけはかっこいいものがあります。しかし，現実にパソコン通信をしているというと，相変わらず「暗い」だの「内向的」だのと色眼鏡で見られることも珍しくありません。

確かに，そういった部分がまったくないと否定はしませんが，こういった一律的な断定は，日本人の大衆やマスコミが大好きな「統一見解」というやつなので，なにもパソコン通信についてだけに見られたことではありません(ちょっと前の有害マンガの焚書騒ぎなどがいい例ですね)。とはいえ,一部のそういった印象を与えがちな(実際にはそうでないことも多い)人によって，パソコン通信というものが，その楽しみを知らないものの興味をもっている人に対してまで悪印象を与えるようなことは，パソコン通信を愛するひとりとして，断じて許せません。

そこで今回は，X68000ユーザー向けに手軽で楽しい通信をより理想的な知識や態度で臨むため，ちょっと偉そうにウンチクを書いてみたいと思います。より楽しいコミュニケーションを求めて，私が適当に身につけた我流の理論を参考にしてもらえれば，嬉しいかぎりです。

パソコン通信は，その歴史や性格から最も新しい，双方向のコミュニケーションメディアです。そこには，参加する人それぞれの道が必ず存在しています。それを見つけることで，パソコン通信の世界へより自然に入っていくことができるのです。その道を探していくことにしましょう。

■

言葉の魔術

■

パソコン通信とは，言葉のやりとりだけで，ホストシステムの空間や時間を共有する人とコミュニケーションを取ることができます。これは，たいへん便利な反面，言葉だけですべてを判断されてしまう怖いところでもあります。困ったことにその言葉すらきちんと読み取れないで，無用なトラブルを招いたりする人もいたりします。同じ文字でも書き手と読み手でニュアンスが異なると，致命的な拡大解釈をされてしまい，噂が独り歩きを始め，収拾がつかなくなるような事態が起こることも珍しくありません。こういった部分はある意味で，社会の縮図的な様相と捉えることもできます。そう，パソコン通信とは社会なのです。それも極めて相手の情報が乏しい，言葉だけでできた不思議な社会なのです。

その中で自分を保ち，目的であるコミュニケーションを維持していくためには，言葉に気を配って配りすぎることはありません。それは単なる言葉使いの問題から，書き込みに使う文字まで，よく考えたうえで自分の言葉を伝えていくことです。ここでは，言葉しかないために一事が万事という法則が成立していることを忘れてはいけません。

たとえば,パソコン通信を示す言葉で「ネット」という表現があります。パソコン通信のホストには，よく「○×ネット」というような名前が多くついていることは，皆さんご存じのことでしょう。それだけの理由ではないのですが，パソコン通信をする人をネットワーカーと呼ぶことも，それなりに定着しています。ネットワーカーというのは，ネットワークにアクセスしている人のことですから，ここで状況的な推理を働かせ「○×ネット」がネットワークだと思う人が少なからずいます。しかしこれは誤解なのです。

もう少し詳しく説明するとネットワークというのは，電話回線のように特定の経路以外の複数のルートで，複数の端子が結合しているものです。ネットとは網のことですから，縦横に伝達のラインがあるというイメージで捉えるとよいでしょう。対して通常のパソコン通信は，いくら回線数が多くても，ホストコンピュータに，参加者のコンピュータがただぶら下がっているだけです。そのため，通常はこういったパソコン通信のホストシステムのことは，BBS(BulletinBoardService)と呼ばれています。本来のBBSはホストシステムの中のサービスである電子掲示板のみを指す言葉ですが，これがサービスの中心であるところが多いため，雑誌などではBBSという表記が一般的なようです。

こういった言葉の場合は，誤用のされ方

がワンパターンなので，わりと好意的に解釈してもらえます。しかし，自分勝手な略称や造語は，ともすればややこしいことになりがちです。フロッピーディスクのつもりで「ディスク」とだけ書いたりすると，話が通じないので相手にされなかったり，きつい調子の非難をあびることもあります。さらに外字などを使った書き込みは，よっぽどのことがないかぎり禁じ手です。

X68000では表示できても，ほかの端末のユーザーに読めないようでは，メッセージの存在価値はないに等しい状態といえるでしょう。実際には，端末に依存する文字としては，X68000で読めないようなNEC系の特殊な文字に接する場合が多いと思われるので，あえてそういう真似をすることのないように，気をつければよいだけかもしれません。

結局，パソコン通信ではしゃべるわけではないのですが「口は禍のもと」という言葉は覚えておいたほうがいいようです。経験的に真面目な意見交換の場になればなるほど，こういった些細なミスの反響は大きくなる傾向があるように見えます。

幻想と現実

また，ときどき見かけるのですが，通信を始めると，便利ですごいツールが山のように手に入って，CGや音楽データでハードディスクが満杯になるという，幻想なのか希望なのか，ちょっと判断しかねる確信に凝り固まった人がいたりします。決まり文句は「X68000のフリーソフトウェアがいっぱいあって，盛んなネットを教えてください」というのがお約束ですが，私はこういう質問にはあまり答えないようにしていますし，事実答えられません。通信とはそういった都合のよいソフトや，データの入手先ではないと思うからです。

ここでハッキリさせておきたいのですが，パソコン通信というものは，コミュニケーションの手段なのです。プログラムをする人の通信スタイルとして，フリーソフトウェアを提供することによるコミュニケーションがあるのであって，それを食い散らかすようにダウンロードするだけでは，そこからはなにも生まれてはきません。使用した結果をレポートするなり，感想を書くなりして，作者とのコミュニケーションを図るのが，あくまでも理想なのです。

結局は通常のボードやSIGへの書き込みとか，ソフトやデータの感想などのコミュニケーションが広がることで，自分にないもの(精神的だったり，具体的なものだったりします)を，親しい相手からもらうことができるかもしれません。それが，少しずつ積み重なった結果として，山のようなツールやデータが手元に残ることがあったとしても，それはあくまで副産物なのです。本当に大事なものはそれまで培ってきたコミュニケーションであり，より深い相手との信頼関係なのです。

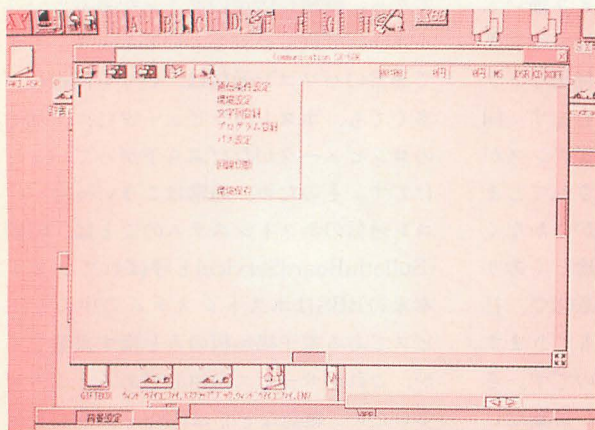
しかしながら，現実を見まわすと商業BBSでは，金銭を払ってサービスを受けるシステムの関係上，規約(禁則事項)というものがあって，その事項に反しないかぎりは原則的になにをやってもいいことになっているところもあります。つまり，書き込みせずにダウンロードするだけ，というコミュニケーションなしの通信スタイルが許可されているところが存在しているわけです。

ここではコミュニケーションを土台に成立していながら，結局は誰もそれが義務だとは決められないという，パソコン通信の複雑な側面が見えてきます。こういった商業BBSの存在価値を否定するつもりは毛頭ありませんが，やはり受け身だけの通信にとどまっているというのは，せっかくの舞台がもったいないような気がするのです。

実はこの問題は，別にX68000ユーザーにかぎったことではなく，もっと根の深い問題だと思います。私自身，他人に説教じみたことをいうほど，立派な行動をとっているわけではありませんし，常に自分の都合のいい方向へ流されてしまっています。あえて文章に書くことでなにかが変われば，と思っているのですが，やっぱりこれが現実というものなのかもしれません。

貧弱な装備

こういった，実際のパソコン通信上でのことだけでなく，X68000では通信を始める前の段階も，かなり難題のようです。モデムは，基本的にX68000で使えないというようなものはないはずなので，速度や値段とかメーカーで選んで買ってしまえばよいのです。しかし，通信ソフトはそういうわけにはいきません。

そもそも通信をまったくやったことのない初心者が入手可能なソフトは，パッケージソフトとして「CommunicationPRO-68K」(SX版含む)と，「たーみのる2」があるだけで，あとはフリーソフトウェアのディスクつきの本に，何本か収録されている程度というところでしょう。これでも以前よりはマシにはなっていますが，NECの国民機と比べたら市販ソフトにしろフリーソフトウェアにしろ，選択の幅に関して比べものになりません。

まずは，通信ソフトの入手(写真右)。そのあとは，実際に試しながら自分なりのツールを見つけてほしい

また，それらのマニュアル本についても，初心者に厳しい状況は同じです。

しかも，通信ソフトは使用目的と操作感で，とことん自分に合ったものを選ばないと不幸になります。そういう意味では通信環境に妥協は許されません。そういわれても情報は少ないし選択の幅も狭い，といいところなしの状態では，おのずと方法はかぎられてきます。とりあえずどれかを購入し，通信することで環境の情報やテクニックを覚え，より自分に便利な環境に，一歩ずつ近づいていくしか道はないようです。

結局そこには，詳しい人や優しい人が大勢いますから，そういう人たちに適切なアドバイスを受けることになるでしょう。すると，ここでもやはりコミュニケーションが大切になってきます。自分がなにをわかっていないのかを，正しく相手に伝えられないと，どれだけ簡単な問題であろうとも，答えが返ってくることは大変疑わしくなります。人によっては初心者に対しても，こういった基本的なコミュニケーションを常識として求める，厳しい態度をとるような人もいます。

これらの条件は，初心者に敷居が高いような印象を与えますが，それさえ乗り越えてしまえば，そこにはかけがえのないコミュニケーションの世界が広がっているのです。

万難を排してアクセス

実際に通信をすると，最も根本的なところにある，電話代やアクセス料金といった部分を無視することはできません。しかもそれが，パソコン通信に対して最も大きな影響力をもっているのです。これはある種の不可抗力と考えて，あきらめるのがいいでしょう。これに関しては，どんな人も同じ見解を示すわけで，間違っても電話料金をどうにかして払わずにすむような犯罪に手を染めてはいけません。ほかにも寝る時間がなくなったとか，夫婦の仲が悪くなったなどのトラブルを耳にしますが，残念ながら自分で解決してほしいと思います。

とまあ長くなりましたが，正直なところOh!Xで通信の話を書く機会がなかったので，あれも書きたい，これも書きたいと思っていたら，ついついとりとめもなく，中途半端な話の山になってしまいました。

また機会があれば，もっと系統立てて，初心者からパワーアクセスユーザーまでを網羅した，通信のガイドを書いてみたいと思います。はっきりと約束はできませんが，私自身も楽しみにしてチャンスを待つことにします。

さて，これでやっと原稿も終わったから，今晩もアクセスしてから寝るとしましょう。おや，空が明るくなってきたぞ。また今日も徹夜かあ，しょうがないな(笑)。

愛のあるユニークで豊かなお部屋

伊澁見あきら

全体のコンセプトは，すべての作業を同じポジションで可能にすることです。これは不精なだけなんですが。まあ，必要は発明の母とでもいっておきましょう(いわないって)。

●机

奥行90cm×幅180cmという，畳1枚に匹敵する巨大な机です。座椅子で向かうとちょうどよいあんばいです。東急ハンズで厚さ3cmの合板と鉄パイプの足を買ってきて組み立てました。幅よりも奥行があることで，背面に余裕をもたせても手前の作業スペースが十分確保できます。あまり手前を広げると，フロッピーの出し入れがしにくくなるので，ほどほどの場所に置いてあります。

●X68000

X68000 XVIです。夏が終われば買い替えて1年です。当時異常なほどの低価格で入手したのが自慢でしたが，高橋氏にその上をいかれてしまい自慢できなくなってしまいました。4Mバイトメモリと HAL研のスキャナボードが差してあります。

●モニタ

もうロートルといえるNECマルチスキャンディスプレイ。3モードなので本体の色を除けばX68000 XVIとの相性はなかなかです。画像の発色など，個人的に純正より上をいくと思っています。

●ハードディスク

ロジテックのMacintosh用の240Mバイトハードディスクです。ドライブ内蔵キャッシュのおかげで通常はかなり速いのですが，起動がとろいのでハードディスクの電源を入れてから10まで数えて，X68000 XVIの電源を入れる必要があります。

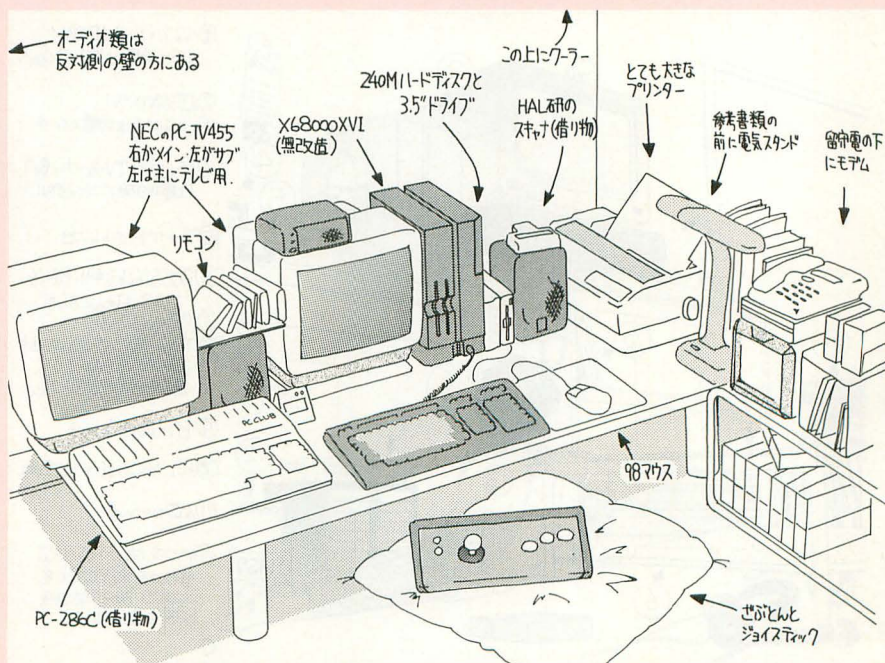
●3.5″FDD

ジャンクのドライブをジャンク屋で買った製品見本用の空ケースに入れたものです。前から見るかぎりは市販品ですね。

●プリンタ

ほとんどドキュメント打ち出し用のエプソンのインクジェットプリンタ。場所をとらない最新型が欲しいところ。

●モデム

9600bpsがブームになる前のオムロンの9600bpsモデム。1991年の暮れに購入。当時，中古でも高かったけど元はとったと確信しています。

川原由唯

部屋にいるときは必ずなにか音楽かけます。最近はCD買うのが趣味になってきてるくらい。チューナーはJ-WAVEに固定しちゃってるしね。

●X68000 ACE-HD（グレー）

内蔵20Mバイトハードディスクは壊れてしまったので，ケースを開けて取り出してしまいました。メインメモリは現在6Mバイト。ちょっと前までは2M＋2M＋4M＝8Mバイト積んでたんだけど，SCSIボードを差してしまったんで，行き場を失った2Mバイトのボードを泣く泣くＪ氏に売却。あとは数値演算プロセッサボード。

●アイテック ITX640

SASIの40Mバイトハードディスク。アクセスするたびにきゅるきゅる唸るので夜中には使えない。

●CZ-6VTI カラーイメージユニット

買ったけどほとんど使った記憶がないなあ。なんかそういう人多いみたいで。DōGAとかやる人だと使うのかな。

●TS-3XR 3.5″FDD

98ノートやUNIXなんかとファイルをやりとりするときに使ってます。

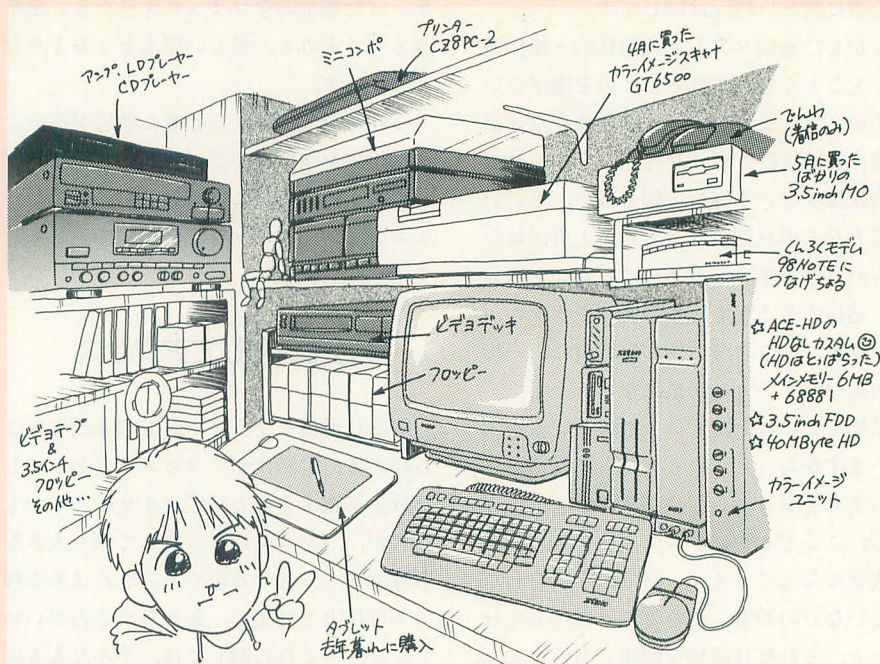
●SC-510C タブレット

タブレットとしては最もオーソドックスなタイプかな。スタイラスペンは好みでストロークのあるやつを使ってます。

●HS10R2 ハンディスキャナ

据置型のスキャナを買ったので隠居の身。

●GT-6500 カラーイメージスキャナ

関係者の知人を通したので，発売されたばかりだけど安く手に入れられた。RS-232Cで接続して使ってます。X68000にもSCSI接続できないかな。

●RMO-S350 3.5″MOドライブ

環境が窮屈になり身動きとれなかったので，衝動的に買ってしまった（いつもにこにこカード払い）。メディアの価格も手ごろだし，便利。

●CZ-8PC2 熱転写式プリンタ

最近はほとんど使ってないなあ。

●PC-9801 NS/E

メモ代わり，ノート代わり。外界との接点。40Mバイト HD＋4Mバイト RAM。

●MD96FB5V 9600モデム

おもにPC-9801 NS/Eにつないでます。

高橋哲史

最近，X68000 ACE-HDを友達に売って友達からXVIを買ったので（X68000でつなぐ友達の輪），それにともないリムーバブルHDの導入などを行って，一見豪華なシステムに見えるようになっています。しかしこの背後には迫りくる家賃と生活費が……ま，いいか。

ミキサーを買ってから曲作りはかなり楽になりました（ちなみにこれも友達から１万円の破格値で買いました）。サコムのMIDIボードから２つのMIDI OUTがとれるため，それほど信号の遅れを気にしなくてもすむ構成になっていますが，やはりそろそろMIDIパッチベイが欲しいところですね（そんな金ないけど）。メロディを作っているときはだいたいD-10を使いますが，場合によってはカシオのポータブルキーボードMT-140も利用しています。

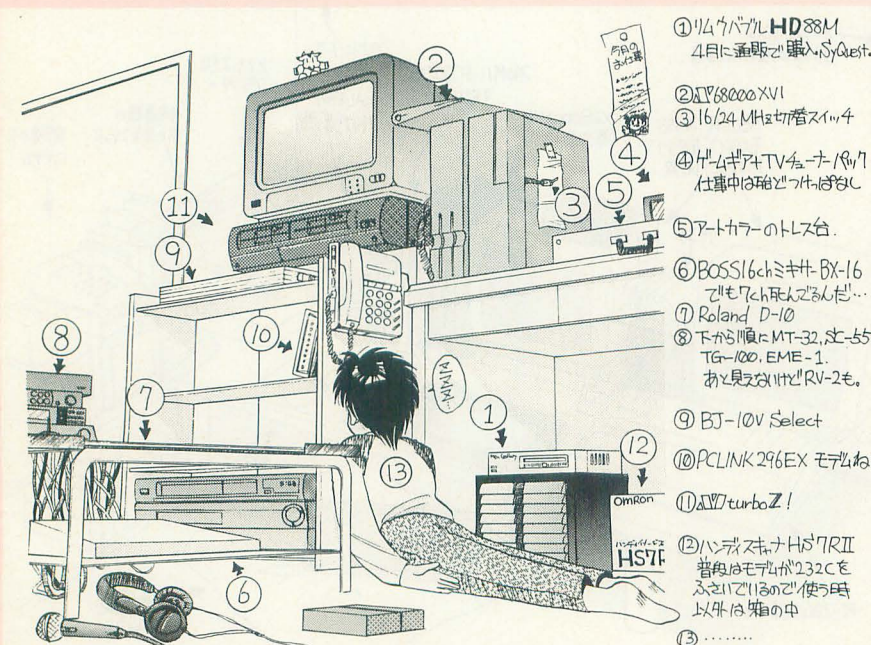
なぜかというと私は男のくせに異様に手が小さく，普通の鍵盤だと１オクターブがやっと届くか届かないかくらいだったりするからです（ミニ鍵盤ならC4からE5くらいまでは届くんですが）。

あ，あとイラスト中には描き忘れてしまいましたが，YAMAHAのQY10も所有しています。最近ほとんど使ってないんですけど。まあこれは，ずーっと家にいるもんで，外で曲を作る機会ってあんまりないんです。

絵を描くときはたいていハンディスキャナで取り込んで修正……という手順を踏んでいます。最近98マウス用アダプタを作ってもらったので（以前Oh!Xで紹介されていた自作ものですね）重宝して使っています。やはり98マウスは種類も豊富なだけに自分に合うものがあって嬉しいですね。

リムーバブルHDは将来MacintoshやAMIGAを買ったときにも，メディアを交換すればそれぞれの機種で使える！　お得だ！　と思って購入しました。Macintoshで原稿描いて「りむーばぶー」で入稿なんてできたらかっこいいなあ………とか思うんですが，とうぶん先の話になりそうなのが悲しいところです。

余談ですが，我が家にある３台のビデオデッキのうち１台は「ある番組」を録るために新規購入されたもので，通称「ウゴウゴデッキ」と呼ばれています（笑）。

吾輩はX68000である
［第25回］

CPUとDMACの共和制

Izumi Daisuke
泉　大介

メモリアクセスの絶対王制から共和制へ
それを，CPUとともに担うのがDMACだ
今回はこのDMACを扱ってみる

illustration：H.Yamada

前々回，前回の2回にわたり，メモリというデバイスについてお話してきた。諸兄はすでに，アドレスバスに適当なアドレスを載せるとメモリチップからデータがデータバスに出力されること，そして，アドレスバスとデータバスに適当なデータを載せてメモリのチップのR/$\overline{\mathrm{W}}$ピンを$\overline{\mathrm{W}}$にすると，そのアドレスに指定したデータが書き込まれることをご存じのはずである。今回はこの話を敷衍し，もう一歩進めた内容をお届けしたいと思う。

◆メモリ間データ転送のプログラム

ご存じのようにMC68000には，メモリからメモリへデータを転送する命令が用意されている。たとえば，

move.w　$100000,$200000

というやつである。これはデータ転送を行うアドレスを直接指定する方法で，この命令を実行すると100000_Hに格納されているワードデータが，アドレス200000_Hにコピーされる。

転送するデータが入っているアドレスやデータを転送する先のアドレスは，アドレスレジスタを使って指定することもでき，これは，

move.w　(a0),(a1)

となる。この例では，A0.Lに格納されているデータが転送元のアドレス，A1.Lに格納されているデータが転送先のアドレスとなり，アドレスA0.Lに格納されているワードデータがアドレスA1.Lへコピーされる。

さらにこれを発展させたものが，

move.w　(a0)＋,(a1)＋

というやつであり，データの転送後アドレスレジスタが自動的に更新されるという特長がある。この例ではアドレスA0.Lに格納されているデータがアドレスA1.Lにコピーされたあと，転送したデータのサイズ分だけ，つまりA0.LとA1.Lがそれぞれ自動的に2バイト大きくなるのである。となれば，100バイトのデータを転送するのは実に簡単なことで，

図1　データ転送のプログラム

1)　200000Hから200100Hへデータを転送するプログラム

```
-z0=200000
-an .z0
00200000        movea.l #$200000,a0     * 転送元アドレス
00200006        movea.l #$200100,a1     * 転送先アドレス
0020000C        moveq   #50-1,d0        * 繰り返し回数
     loop:
0020000E        move.w  (a0)+,(a1)+     * ワードデータ転送
     †          dbra    d0,loop
00200010        dbra    d0,.z0+$0e      * 繰り返し
00200014        _exit
00200016
```

2)　200100Hをダンプしてみると

```
-d 200100
00200100  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
00200110  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200120  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
00200130  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200140  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200150  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
00200160  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200170  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
```

3)　プログラムを実行

```
-g=200000
program terminated normally
```

4)　再び200100Hをダンプしてみる

```
-d 200100
00200100  207C 0020 0000 227C 0020 0100 7031 32D8     |. .."|. ..p12ﾘ
00200110  51C8 FFFC FF00 0000 FFFF 0000 FFFF 0000    Qﾈ..............
00200120  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
00200130  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200140  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200150  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
00200160  FFFF 0000 FFFF 0000 FFFF 0000 FFFF 0000    ................
00200170  0000 FFFF 0000 FFFF 0000 FFFF 0000 FFFF    ................
```

5)　ちなみにこれを逆アセンブルすると

```
-l 200100
 00200100       movea.l #$00200000,A0
 00200106       movea.l #$00200100,A1
 0020010C       moveq   #$31,D0
 0020010E       move.w  (A0)+,(A1)+
 00200110       dbf     D0,$0020010E
 00200114       _EXIT
 00200116       ori.b   #$FF,D0
 0020011A       ori.b   #$FF,D0
```

move.w　(a0)+,(a1)+

という命令を延々50回繰り返せばいい。プログラムにすると，図1のようになるだろう。

図1-1は，200000$_H$から200100$_H$へ100バイトのデータ転送を行うプログラムである。最初200100$_H$には図1-2のようになにもデータが入っていないが，このプログラムを実行すると図1-4のようにデータが転送される。200000$_H$に入っているのは図1-1のプログラムなので，200100$_H$へはこのプログラムが転送されているはずである。実際に200100$_H$以降を逆アセンブルしてみると，図1-5のようになる。この逆アセンブルリストを眺めて，図1-1で，

dbra　d0,.z0+$0e

となっていたdbra命令のジャンプ先(.z0+$0e=20000E$_H$)が，

dbra　d0,$20010e

と20010E$_H$になっているのを奇妙に思う方がいらっしゃるかもしれない。これは，dbra命令がジャンプ先を相対アドレスで指定するようになっているためである。つまり実際のdbra命令のジャンプ先は，「dbra命令からnバイト前」あるいは「nバイト後」という形式で指定されているのだ。そんな表記をされても，16進の加減乗除にうとい人間にはそのジャンプ先がどこなのか一瞥してわからないため，デバッガは親切にもジャンプ先をアドレスで表示しているわけである。

◆転送手順を追ってみると

図1のプログラムを実行し，その転送手順を追ってみたのが図2である。move命令によるデータ転送と，再びmove命令を実行するためにジャンプするdbra命令が交互に実行されているのを確認していただけるだろう。前回までの知識を利用し，もう少しハードウェアのレベルに降りてこの様子を眺めると，次のような手順がCPUによって実行されていることを想像できると思う。

1)　アドレスバスに20000E$_H$を載せる
2)　出てきたデータをデコードする
3)　「move.w (a0)+,(a1)+」命令だとわかる
4)　A0.Lをアドレスバスに載せる
5)　出てきたデータを取り込む
6)　A1.Lをアドレスバスに，取り込んだデータをデータバスに載せ，データを書き込む
7)　A0.L，A1.Lを2バイト大きくする
8)　アドレスバスに200010$_H$を載せる
9)　出てきたデータをデコードする
10)　「dbra d0,〜」命令だとわかる
11)　アドレスバスに200012$_H$を載せる
12)　出てきたオフセットを取り込む
13)　D0.Wをひとつ小さくする
14)　−1にならなければ，オフセットからジャンプ先を計算する
15)　1)から繰り返す

途中はしょったところもあるし，CPUの動作に即してわかりやすいように書き直したところもあるが，図2で行われていることはおおむね上のようになる。実際にデータ転送を行っているのは4)〜7)の部分だが，4)〜7)を繰り返し実行するためだけに1)〜3)，8)〜15)のこれだけの手順が必要なのである。

もし，4)〜7)だけを13)の条件つきで実行できれば，データの転送速度を大幅にアップすることができるのではないだろうか。少なくとも手順の数は1/3程度にまで減る。さらに，1)，8)，11)のような「プログラムを取り出すため」のアドレスバスの操作はなくなり，取り込んだ命令をデコードする手間も，ジャンプ先を計算する手間もない。単に，データを取り出すアドレスとそれを書き込むアドレスを交互にアドレスバスに載せ，メモリチップにつながるR/$\overline{W}$をパタパタ切り替えてやるだけで，データを望みの場所から望みの場所へと転送できるのであ

図2　転送手順を追いかけてみる

```
1) move命令の実行

-t
PC=0020000E USP=000EF364 SSP=000BD9B4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000031 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00200000 00200100 00000000 00000000  00000000 00000000 00000000 000EF364
move.w  (A0)+,(A1)+                    :00200000(207C)


2) dbra命令の実行

-t
PC=00200010 USP=000EF364 SSP=000BD9B4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000031 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00200002 00200102 00000000 00000000  00000000 00000000 00000000 000EF364
dbf     D0,$0020000E


3) move命令の実行

-t
PC=0020000E USP=000EF364 SSP=000BD9B4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000030 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00200002 00200102 00000000 00000000  00000000 00000000 00000000 000EF364
move.w  (A0)+,(A1)+                    :00200002(0020)


4) dbra命令の実行

-t
PC=00200010 USP=000EF364 SSP=000BD9B4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000030 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00200004 00200104 00000000 00000000  00000000 00000000 00000000 000EF364
dbf     D0,$0020000E


5) move命令の実行

-t
PC=0020000E USP=000EF364 SSP=000BD9B4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0000002F 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00200004 00200104 00000000 00000000  00000000 00000000 00000000 000EF364
move.w  (A0)+,(A1)+                    :00200004(0000)


6) dbra命令の実行

-t
PC=00200010 USP=000EF364 SSP=000BD9B4 SR=8004 X:0  N:0  Z:1  V:0  C:0
D  0000002F 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00200006 00200106 00000000 00000000  00000000 00000000 00000000 000EF364
dbf     D0,$0020000E
```

る。かなりの高速化が期待できることは想像に難くない。

そして，実際にそう考えた輩がいたのである。その結果誕生したのが，DMA(Direct Memory Access)という手法であり，DMAC(DMA Controller)と呼ばれるチップなのである。

◆バスを乗っ取る

前回まで説明したように，アドレスバスはデータを取り出すメモリアドレスの通り道，データバスはメモリから取り出される，あるいはメモリに書き込まれるデータの通り道である。コンピュータの仕事とは，煎じ詰めれば結局メモリからレジスタへ，レジスタからメモリへ，あるいはメモリからメモリへとデータを移動させることにほかならない。この結果さまざまなデータがバスの中を駆け巡ることになるのだが，バスにどんなデータを載せるのかを命令しているのはCPUだけである。CPUは専制君主であり，メモリはその重臣にたとえられよう。

CPUによる絶対王制下では，君主と臣下が膝突き合わせて国策を検討するということがない。王様は玉座に奉られて別室にあらせられ，臣下の者たちは施設されたホットラインを通じて王様のお声を拝聴することができるだけである。王様はホットラインを通じて，「○○番地のデータを届けろ」とか「このデータを○○番地に格納しておけ」とか命令を下す。命令を受けた重臣は，「はい，ただいま」とかいいながら，王様のお膝元に直結しているベルトコンベアにデータを載せたり，王様が自らコンベアに載せられたデータを拝領してしまい込むのである。関連はおわかりだろう。ホットラインがアドレスバス，ベルトコンベアがデータバスである（図3）。

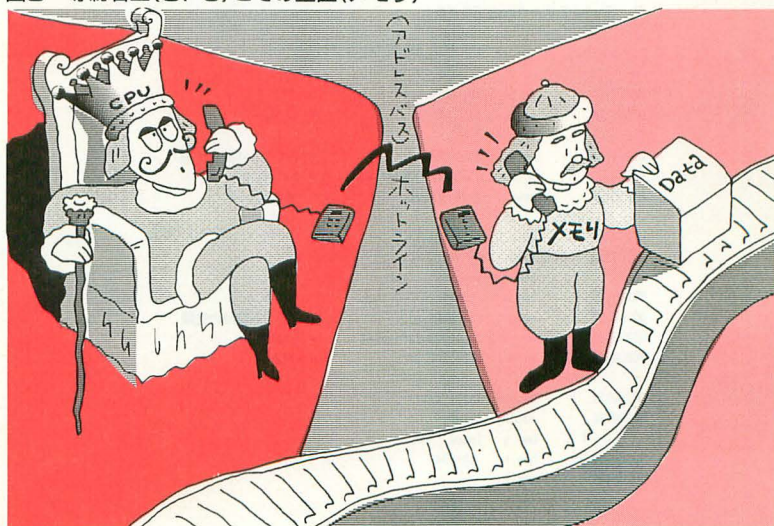
このシステムがうまく動くのは，命令を出す者がCPUしかないからである。もしホットラインに誰かが割り込んでほかのアドレスからデータを取り出そうとしたり，データをほかのアドレスに書き込もうとしたりすると，システムは間違いなく混乱をきたす。もちろん，CPUがアドレスバスやデータバスを使用していない狭間にうまくもぐり込むことができれば初期の目的を達成することができるかもしれない。しかしながら，この第3列がホットラインを使用しようとしたときに王様が使っていないという保証はないのだ。電話なら「先に話をしていた者の勝ち」という排他制御があるのだが，残念ながらこのホットラインにはそのような仕組みがない。両者が鉢合わせすると，1本しかないホットラインの中で命令が交錯し，事態は紛糾することだろう。

では，王様が使っているのとは別の方法でメモリからデータを受け取ったり，メモリにデータを書き込むことは可能だろうか。残念ながら，その手段はこのホットラインとベルトコンベアを使うよりほかにはないようである。というのも，これ以外の方法で命令を受け取る手段をメモリがもっていないからだ。データの高速転送を行うにはホットラインとベルトコンベアが必要なのだが，それは鉢合わせの危険なくして利用することはできない。それが，第3列たるDMAの置かれている立場である。

このやっかいなジレンマを解決する手段はひとつしかない。絶対君主制から共和制への移行である。データが高速に転送されることはCPUにとっても願ったりかなったりなので，CPUは絶対君主の座を降り，DMAと協調して2つのバスを使い回せばいい。まあ，寡頭政治といってもいいだろう。そのためにはまず，現在CPUがしっかりと握っているバスの制御権をDMAにも分け与えることだ。普段はCPUがバスの制御権をもち，データを読み込んでプログラムの実行を行う。ここまではいままでと同じ。新たにつけ加わった条例は，データの高速転送が必要な局面ではCPUはバスを解放しDMAが自由に使えるようにしてやるのである。逆のいい方をすれば，必要なときにDMAはバスを乗っ取れるということになる。

では，実際にどうすればCPUはバスを明け渡すことが(DMAがバスを乗っ取ることが)できるのだろうか。単純に考えれば，CPUがアドレスバスやデータバスとの結線を切れば，以後DMAは自由にこれを制御することができそうである。もちろんハンダづけされているリード線をCPUが好き勝手に切断したり接続し直したりすることはできないので，これに代わる方法が必要だ。通常，バスの明け渡しは，CPUのアドレスピンとデータピンの抵抗値を十分大きくすることによって実現されている。俗にいうハイインピーダンスというやつだ。

ハイインピーダンスにすることがどうしてバスを明け渡したことになるのかというと，例のオームの法則である。I＝E/Rという式が示すように，回路に流れる電流は電圧が高いほど，抵抗が小さいほど大きくなる。アドレスバスやデータバスにかかる電圧は一定なので，ハイインピーダンスになり抵抗値が大きくなると，アドレスピンやデータピンにはほとんど電流が流れない。つまり，

図3　専制君主(CPU)とその重臣(メモリ)

電気的に絶縁され，リード線を切られたのと同様の状態になるのである。

こうしてCPUがバスから外れたあとは，DMAの独壇場となる。アドレスとデータをバスに載せR/$\overline{\mathrm{W}}$をパタパタやるという前述の方法を使って，データの高速転送をやってのけることが可能となるわけである。データ転送が終了するとDMAはバスを解放し，CPUに「終わったよ」と告げる。それを合図にCPUは再びメモリからデータを取り出し，デコードし，……といういつもの作業を再開することになる。

◆DMA転送の実際

吾輩には，HD63450というDMACが搭載されている。DMACはその名の由来どおりに「ディー・エム・エー・コントローラ」と読んでもいいし，もっと手短に「ディー・マック」と呼んでもいい。うちの御仁はなにかにつけて形式張るのが好きなので，面倒な前者の呼び方をしている。HD63450は4つのDMAチャンネルをもち，FDDやHDD，AD PCMといった高速なデータ転送を要求されるデバイスとのデータやり取りを受けもっている。またそのデータ転送方法もひととおりではない。このあたりに踏み込むとひと筋縄ではいかなくなるので，今回は諸兄にその威力を体験していただく程度に留めておくことにしたい。

これまでに紹介したMFPやCRTCのように，DMACもさまざまなレジスタをもっている。これらのレジスタに種々のデータを設定することで，DMACはバスを乗っ取って動き始めるのだが，手軽にDMAのパワーを試す程度ならIOCSコールを使って実験してみる程度で十分である。IOCSコール$8A_H$は次のようにデータをセットして使用する。

A1.L　アドレス1
A2.L　アドレス2
D1.B　転送モード
D2.L　転送バイト数

この中で重要なのがD1.Bのモード設定である。これはビットごとに次のような意味をもっている。

第7ビット　データの転送方向
第3,2ビット　A1.Lの増減制御
第1,0ビット　A2.Lの増減制御

データの転送方向は，0ならアドレスA1からアドレスA2への転送を，1ならその逆を意味する。増減制御は2ビットで指定し，00なら変化なし，01なら増加，10なら減少である。したがって，アドレスA1からアドレスA2へレジスタの値を大きくしながら転送したいなら，つまり，「move.b (a1)＋,(a2)＋」という動作をさせたいなら，D1.Bは00000101_Bとなる。

これを踏まえたうえで，図4をご覧いただきたい。図4には2つのプログラムを用意してある。1)はDMAを使ったデータ転送，2)は比較の意味で用意したCPUを使ったデータ転送である。いずれも，テキスト画面を1行スクロールさせるプログラムとなっている。ご存じのようにテキストVRAMは$E00000_H$から始まっており，1バイトが横8つのドットの点灯，消灯を意味するようになっている。テキスト画面は1024×1024ドットなので1ドット下は$E00080_H$になり，1行下(16ドット下)は$E00800_H$となる。冒頭でこの2つのアドレスをA1.LとA2.Lにセットしているのを確認していただきたい。転送ドット数は512ライン分，すなわち，80_H×512バイトである。通常画面に表示されている白い色は2つのプレーンに同じものを表示して実現されているので，ここではさらに$E20000_H$から始まる次のプレーンに対しても同じ処理を繰り返している。

2つのプログラムを入力したら，まずDMA版，続いてCPU版を実行してみていただきたい。かなりスピードが違うのを確認していただけることと思う。CPU版がバイト転送になっているのがズルイ，と思う方がいらっしゃるかもしれないが，IOCSコール$8A_H$もデータのバイト転送を行っているので条件は同じである。この結果を見てもまだ，王制復古を望む方は果たしていらっしゃるだろうか。

図4　DMAによるデータ転送とCPUによるデータ転送

1)　DMAによるデータ転送プログラム

```
-z0=200000
-an .z0
00200000        movea.l #$e00000,a1             * テキストプレーン1の先頭
00200006        movea.l #$e00000+128*16,a2      * その1行下
0020000C        move.b  #_10000101,d1           * (a2)+,(a1)+
00200010        move.l  #128*512,d2             * 1画面分
00200016        moveq   #$8a,d0                 * _dmamove
00200018        trap    #15
0020001A        movea.l #$e20000,a1             * テキストプレーン2の先頭
00200020        movea.l #$e20000+128*16,a2      * その1行下
00200026        move.b  #_10000101,d1           * (a2)+,(a1)+
0020002A        move.l  #128*512,d2             * 1画面分
00200030        moveq   #$8a,d0                 * _dmamove
00200032        trap    #15
00200034        _exit
```

2)　CPUによるデータ転送プログラム

```
-z0=200100
-an .z0
00200100        movea.l #0,a1
00200106        moveq   #$81,d0                 * _b_super
00200108        trap    #15
0020010A        movea.l #$e00000,a1             * テキストVRAMの先頭
00200110        movea.l #$e00000+128*16,a2      * その1行下
00200116        move.l  #128*500,d0             * 1画面分
        loop:
0020011C        move.b  (a2)+,(a1)+
0020011E        subq.l  #1,d0
        †       bne.s   loop
00200120        bne.s   .z0+$1c
00200122        movea.l #$e20000,a1             * テキストVRAMの先頭
00200128        movea.l #$e20000+128*16,a2      * その1行下
0020012E        move.l  #128*500,d0             * 1画面分
        loop1:
00200134        move.b  (a2)+,(a1)+
00200136        subq.l  #1,d0
        †       bne.s   loop1
00200138        bne.s   .z0+$34
0020013A        _exit
```

(で)のショートプロぱーてぃ――その47

4周年ショートプロ
パソコンは死なない！

Komura Satoshi 古村 聡

いつの間にやら4周年。ここまで続けられたのも，ひとえに読者の皆さんからの投稿の賜物です。今月はなかなか入力のしやすい適当なプログラムが3本ありますので，ぜひ，遊んでみてください。

illustration：T.Takahashi

おかげさまでショートプロも4歳です，ありがとうございます。今年は忘れないでよかった。それにしても，「4周年おめでとーっ」って投稿があるかと思ってたんだけどな，なかったんだよね。う～ん，あったら絶対採用にしようと思っていたのに。ちぇ(もう，過ぎたからなんとでもいっちゃうぞ，わし)。

それにしても4周年ってことで改めて考えてみるんですが，ここ最近のパソコン，それもX68000以外の機種のあり方って間違ってると思いませんか!? たとえば最近のなんちゃらVではCD-ROMを買ってきてマルチメディアなんちゃらを見られる。なるほど。でも，そんなのビデオでだってできることでしょ。いままでの「マス」メディアとなんら変わらないんだったらなんのための「パーソナル」コンピュータなんだ!? って気がしませんか？ ましてやビデオカードを入れ替え差し替え，設定変えて動けばラッキィ，なんてそんな楽しみ方，なんの意味があるんだって思うんですよ。

パーソナルである，ということは自分にしかできないことができるっていうこと。まして，パソコンは絵だって音楽だってプログラムだって自分で作れるんですよね。だったら，パソコンで何かを作る，自分の何かを表現する。それこそが，私はパーソナルなコンピュータのある意味だと思うし，正しいあり方だと思うんですよね。プログラムでもお絵描きでも，あるいはパソ通での書き込みでも。

ってことではなはだ勝手ながら，ショートプロぱーてぃは自分で何かを作りたい人，作っちゃう人，そんな人を応援し続けるために続いてしまうのでありました。さあ，ショートプロの5年目スタートだ！

SCROLL3D.BAS

くらいよせまいよ3D

では今月の1本目，宮城県の高木さんによるX-BASIC用3DゲームでSCROLL3D.BASです。どうぞっ。

SCROLL3D.BAS for X68000/030
(要X-BASIC，ジョイスティック)
宮城県 高木大輔

このショートプロでも「壁をよけて進め！」ってタイプのゲームはよくありましたよね。そうそう，ジョイスティックでトンネルみたいな中を通っていくやつ。ものによっては自機の動きに慣性がかかってたりするあれです。このゲームはただの壁よけゲームではありません。なあんと，3Dによる立体的な壁よけゲームなのです！ このゲームはX-BASICのゲームですので，

A>BASIC

でX-BASICを起動し，リスト1を入力し，間違いがないことを確認して，セーブしてからRUNでゲームを始めましょう。

遊び方は簡単。背景がずりずりと動き始めますのでジョイスティックの左右で壁に当たらないように進んでください。で，壁に当たるとゲームオーバーです。ジョイスティックのボタンで再スタートします。

おお，このリストの短さなのに3次元で壁がニュルニュルと動く動く！ しかもX68000の10MHzで十分遊べる(っていうかそれより高速なモードだとゲームとしては難しすぎる)！

このプログラムは壁の動きにパレットアニメーションを使っているんですが，このパレットアニメーションってご存じですか？ たとえばですね，矢印のキャラクタを左から右に動かしたいとしますね。このときにキャラクタを使うなら，

1) 一番左に矢印を表示
2) その右に矢印を表示して
3) 一番左の矢印は消す
4) さらにその右に表示

を繰り返すとか，あるいはスプライトだったら，

1) 矢印のキャラクタをスプライトに定義
2) 一番左に矢印を表示
3) その右に表示
4) そのまた右に表示

とまあ，スプライトだから消す必要はないんですけど，とにかくそうやってアニメーションするわけです。

で，パレットアニメーションっていうのはこいつらとは違って，画面にすべてのパターン，

→→→→→→→→→→

などを描いておくのです。ただし，これらのパレットを左から1，2，3番……と1つひとつ違う色にしておくんですね。で，まず，pallet()関数で使っているパレットを背景と同じ色にしておきます。それから1番の色をつけて，1番を消して2番をつける……と順番に色をつけていくとあーら不思議，動いているように見えるじゃないですか。これをパレットアニメーションというのです。え，別に不思議でもなんでもないって？ ネオンサインといっしょじゃないかって，そうなんですけどね。

で，何が有利かってこのパレットアニメーション，動きながら描き直すわけじゃないから，(見かけの)スピードが速いし，スプライトみたいに大きさが決まっていないので大きなものを描くにはとても有利なんですね。ただし，使う色数が減ってしまうことやあらかじめ絵を描いてしまうので，あまり複雑な動きはしにくい，という問題

があります。このへんってプロでも使い分けに悩むところなんだそうですけどね。そういう意味では，パレットアニメを使って3Dゲームを作った高木さんの判断はたいしたもんだと思います。

ところで340行と360行を有効にすると派手になります。壁の色は140行のHSV関数の値を変えることで変更できるそうなんでやってみてくださいね。

並んで並んでインデント

続いて2本目。千葉県の堀さんのプログラムで，X-BASICテキストの字下げプログラムIND.BASです。どうぞ〜。

IND.BAS for X68000/030

(要X-BASIC)

千葉県　堀貴志

X-BASICテキストの字下げを行うプログラムです。リスト中にfor〜nextやfunc〜endfuncなどがあった場合，その間を一定の間隔で空白を入れてリストを見やすくします。これもX-BASIC用のプログラムですから，X-BASICを起動後リスト2を入力して，RUNで起動できます。

使い方ですが，まずSAVE@でセーブした行番号のないプログラムファイルを用意してください。

このプログラムをRUNさせるとファイル名を聞いてきますので，〜.BASまで省略しないでSAVE@でセーブしたファイルの名前を入れてください。

次に変換後にセーブしたいファイルの名前を同じように省略せずに入力します。元のプログラムと同じ場合には，何も入れないでOKです。

最後に字下げ幅を聞いてきますので，数字でスペースを何個入れるかを入力してください。デフォルトは2になっています。

それが終わると1行1行プログラムがどのようにリストを変えていっているか表示していってくれます。表示が終わると新しく字下げされたプログラムが完成しているはずです。

X-BASICプログラムによるX-BASICプログラムのための字下げプログラムですね。実はこのプログラムの作者の堀さん，オリジナルのind.basから改良に改良を重ねたのか，ind1，ind2……といちばん最新のバージョンのind8.basまで7本もプログラムを送ってきてくださったんですよ。あたしゃ感動してしまいましたですよ，はい(あ，ちなみに掲載されているのは最新バージョンのind8.basを元にしたものです)。

動かないよと思う前に(10)

★「関数は定義されていません」

「関数は定義されていません」というメッセージは文字どおり，関数がなかった場合に出ます。では，なぜその関数がないのかというと，だいたい次の3つのうちのどれかが原因という場合が，ほとんどだと思います。

ひとつは呼び出し関数名をタイプミスしている。そりゃあ，本当はfoo()を呼ばなくちゃいけないのにhoo()を呼んでしまったら，その関数はありませんよね。

2つ目は関数の定義側をタイプミスしてしまった場合。1番目の逆でfoo()を呼びたいのに，その関数が定義されているはずのところで間違えていたら，ひとつめと同じってわけですね。

それから3つめ。X-BASICには外部関数というものが作れて，いろいろな関数をあとからつけ加えることができるんでしたよね。てことは，逆に必要な関数が外れてしまっていることも考えられる，ということなんです。

たとえばグラフィック関係の関数を使っているのにそれ用の関数が外されてしまっていたり，あるいはZ-MUSIC関係の命令を使うのに，MUSICZ.FNCを組み込むのを忘れていたりとかね。この外部変数を組み込む定義をしているのは，BASICのディレクトリの中にあるBASIC.CNFというファイルです。このエラーが出たときには必要な機能が，

FUNC＝〜

でちゃんと組み込まれているか確認してみてください。

リスト1　SCROLL3D.BAS

```
 10 /*  Scroll for X68000
 20 /*
 30 /*         by D,Takagi   1993.3
 40 /*
 50 /* 初期設定
 60 screen 0,1,1,1
 70 int   d1(8),d2(8),mp(8),ms(359)
 80 int   pt,pa,mx,dx,sc,tn,i,co=128
 90 float d3(8)
100 for i=0 to 359:ms(i)=sin(pi(i/180#))*150:next
110 /* 背景
120 vpage(0):apage(3)
130 for i=0 to 8
140   palet(i+1,hsv(128,31,i*3+7))
150   d1(i)=(160*(9-i))/(12-i)  :d2(i)=255-d1(i)
160   d3(i)=( 1#*(9-i))/(12-i)-1
170   line(0,d1(i),255,d1(i),i+1)
180   line(0,d2(i),255,d2(i),i+1)
190   paint(0,d1(i)-1,i+1)
200   paint(0,d2(i)+1,i+1)
210 next
220 /* Main
230 while 1
240   repeat
250     vpage(pa+9):pa=(pa+1) mod 2:apage(pa)
260     pt=(pt+8) mod 9:sc=sc+1
270     st=stick(1):mx=mx+((st=6)-(st=4))*30
280     tn=(tn+rnd()*30) mod 360:mp(pt)=ms(tn)
290     fill(0,40,255,215,0)
300     for i=0 to 8
310       dx=(mp((pt+i) mod 9)+mx)*d3(i)
320       fill(d1(i)-dx,d1(i),  0,d2(i),i+1)
330       fill(d2(i)-dx,d1(i),255,d2(i),i+1)
340     /*palet(i+1,hsv(co,31,i*3+7))
350     next
360   /*co=(co+1) mod 192
370   until point(127,127)=9
380   locate 7, 7:print"GAME OVER"
390   locate 8,12:print using"@####";"SCORE      ",sc
400   repeat:rnd():until strig(1)<>0
410   /* 再初期化
420   cls:vpage(0):wipe()
430   sc=0:mx=0:tn=0:for i=0 to 8:mp(i)=0:next
440 endwhile
```

リスト2　IND.BAS

```
 10 int n
 20 str fnam1,fnam2,work="qzxep.wgb"
 30 print "テキスト字下げプログラム"
 40 input "ファイル名=";fnam1
 50 input "変換後のファイル名(省略の場合同じファイル)=";fnam2
 60 input "字下げ幅(省略の場合2)=";n        :if n=0 then n=2
 70 if fnam2="" then {
 80   ind(fnam1,work,n):fdelete(fnam1):frename(work,fnam1)
 90 } else {
100   ind(fnam1,fnam2,n)
110 }
120 end
130 /*
140 func ind(fname1;str,fname2;str,wide;int)
150   /* 機能    .. ファイルの字下げを行う
```

きっちり構文解析しているわけでもなくて，instr()関数(文字列中から特定の文字列を見つける関数)でラフにチェックするだけでここまでできてしまうんですね。たいしたものです。

そうそう，このプログラムを使うにあたっていくつか注意があります。

まず，制御文っていうか予約語の名前が入った変数は作らないこと。before_lvとか(笑)。ま，制御文以外なら大丈夫ですけど。それからできるだけ行儀のよいプログラムを書くこと。完全に構文解析してるわけではないので，

for～next : if～then {

などとは段づけがおかしくなってしまうので，実行には害はないですけど書かないでくださいね(え，バグじゃないのかって？仕様です，仕様)。それと，元になるファイル名のプログラムファイルがなかったりするとそのままエラーを表示して止まってしまいます。

作者の堀さん，次はゲームを作るつもりでこのプログラムもそのリストをきれいにするために作ったんだとか。できたら次のプログラムもショートプロに投稿してくださいね。待ってますよ。

では今月最後のプログラム。愛知県の伊藤さんの作品で，簡易メニュー表示プログラム，USERMENU.BASです。どうぞ。

USERMENU.BAS for X68000

(要X-BASIC)

愛知県　伊藤政弘

皆さん，よく使うプログラムやそのオプションってどうしてますか？　私は記憶力もないしキーを打つのも面倒なので，エディタを使うときなんか，必要なオプション設定もひっくるめてE.BATというバッチファイルにして，

A>E

で立ち上がるようにしているんですが，これだとときどきどの文字がどのソフトか忘れてしまったり(なにしろ無理にアルファベット26文字に割り当ててるもので)するんですよね。かといってバッチ式メニューだと数が多いと書くのも大変だし。

ってぇわけで，ここで便利なのがこのプログラムです。

このプログラムはX-BASICで書かれているのでリスト3を入力してUSERMENU.BASという名前でセーブしてください。それから，もうひとつ，

```
 160   /* fname1 .. 字下げ前のファイル名
 170   /* fname2 .. 字下げ後に格納するファイル名
 180   /* wide   ..  字下げ幅(省略の場合2)
 190   str s_cutting[255],s_check[255]
 200   str chk1(16)={
 210     "for " , "{" , "}" , "{" , "}" , "for " ,
 220     "next" , "endfunc" , "func " , "endwhile" ,
 230     "while " , "repeat" , "until " , "case " ,
 240     "default" , "endswitch" , "switch "
 250   }
 260   str chk2(16)={ "next" , "}" , "{" }
 270   /* b_lv .. ファイル書き込み前にプラスするレベル
 280   int b_lv(17)={
 290    +0, 0,-1, 0,-1, 0,-1,-1, 0,-1,
 300    +0,-1, 0,-1,-1,-2, 0, 0 }
 310   /* a_lv .. ファイル書き込み後にプラスするレベル
 320   int a_lv(17)={ 0, 0, 1, 1, 0, 1,
 330    +0, 0, 1, 0, 1, 0, 1, 1,
 340    +1, 0, 2, 0 }
 350   int fp1,fp2,flg1,flg2,j,lv=0
 360   /*
 370   fp1=fopen(fname1,"r")
 380   fp2=fopen(fname2,"c")
 390   /*
 400   freads(s_cutting,fp1)
 410   while feof(fp1)=0
 420    s_cutting=lcut(s_cutting)+chr$(13)+chr$(10)
 430    s_check=cq_cut(s_cutting)  /*注釈とコメントをカット
 440    /*
 450    for j=0 to 16
 460     flg1=instr(1,s_check,chk1(j))
 470     flg2=instr(1,s_check,chk2(j))+n_chk(chk2(j)) /*ここ
 480     if flg1<flg2 and flg1>0 then break
 490    next
 500     /*
 510     lv=lv+b_lv(j)
 520     wprint(adsps(s_cutting,wide,lv),fp2)
 530     lv=lv+a_lv(j)
 540     /*
 550     freads(s_cutting,fp1)
 560   endwhile
 570   fwrites(chr$(26),fp2)
 580   fclose(fp1) : fclose(fp2)
 590 endfunc
 600 /*
 610 func str lcut(string;str)
 620 /* 機能 ..文字列の左の空白とタブを削除して返す
 630 /* string..処理される文字列
 640   str r[1] : char p=1
 650   /*
 660   while 1
 670     r=mid$(string,p,1)
 680     if r=chr$(9) or r=chr$(32) then {
 690       p=p+1
 700     } else {
 710       break
 720     }
 730   endwhile
 740   return(right$(string,strlen(string)-p+1))
 750 endfunc
 760 /*
 770 func  str cq_cut(string;str)
 780 /* 機能..注釈以降と引用符と
 790 /*          その間の文字列を削除して返す
 800 /* string .. 処理される文字列
 810   int q1,q2
 820   while 1
 830     q1=instr(1,string,chr$(34)) /*一番目の引用符
 840     if q1>0 then {
 850       q2=instr(q1+1,string,chr$(34)) /*二番目の引用符
 860       string=left$(string,q1-1)+right$(string,strlen(string)-q2)
 870     } else {
 880       q1=instr(1,string,"/*") /*注釈のチェック
 890       if q1>0 then {
 900         string=left$(string,q1-1)
 910       } else {
 920         break
 930       }
 940     }
 950   endwhile
 960   return(string)
 970 endfunc
 980 /*
 990 func str adsps(string;str,i;int,lv;int)
1000   /* 機能     .. 文字列の左にスペースを入れて返す
1010   /* string .. 処理される文字列
1020   /* i      .. 1回に挿入されるスペースの数
1030   /* lv     .. 何回挿入するか
1040   return(space$(i*lv)+string)
1050 endfunc
1060 /*
1070 func int wprint(string;str,fp;int)
1080   /* 機能     .. 文字列のファイル書き込みと画面出力
1090   /* string .. 処理される文字列
1100   /* fp     .. ファイル番号
1110   fwrites(string,fp)
1120   print string;
1130 endfunc
1140 /*
```

```
1150 func int n_chk(string;str)
1160   /* 機能 .. 文字列が空なら256をそうでなければ0を返す
1170   if string="" then return(256) else return(0)
1180 endfunc
```

cls
echo off
basic menu.bas
menu.bat

という内容でUSEREXEC.BATというファイルを作っておいてください。

それからメニュー定義を書きます。

最初の1行，あるいは$の次の行はメニューの項目名を書いて，そのあとにやりたいことを書きます。それをメニュー項目分，ずらずらと書いていって，USERMENU.DATという名前でセーブしておいてください。例を見たほうが早いですね。

例)

終了
ECHO OFF
CLS
ECHO お疲れ様でした。
$
エディタの起動
ED
$
ビジュアルシェルの起動
VS2
$
フロッピーディスクのフォーマット
FORMAT
$

これで，エディタ，VS2の起動，フォーマットの3つの項目をもつメニューを作ることができます。関連ファイルは，USERMENU.BASと同一ディレクトリに置いてください。あとは，このプログラムを，

A>USEREXEC.BAT

と実行することにより，USERMENU.BASが起動し，USERMENU.DATを読み込みます。そして，メニューの選択により，USERMENU.BATに処理の内容が書き込まれ，USERMENU.BATに実行が移り，選択した処理が実行されます。

ん～，使ってみて思うのは，なんとも実用的で便利なプログラムだっていうこと。必要に迫られて作った他機種用のプログラ

リスト3 USERMENU.BAS

```
  10 /*
  20 /*処理選択システム（アプリケーション起動プログラム）
  30 /*OAMENU.BAS 初期バージョン
  40 /*   programed by 1991. 8.26 名畑 for PC-9801
  50 /*OAMENU.BAS メニュー外部ファイル化
  60 /*    programed by 1992. 2. 1 吉田 for PC-9801
  70 /*OAMENU.BAS タイマー機能組込み
  80 /*    programed by 1992. 4.30 伊藤 for PC-9801
  90 /*USERMENU.BAS  移植初期バージョン
 100 /*    programed by 1992. 6.15 伊藤 for X68000
 110 /*
 120 int position(20),finput,foutput,po,mdat,flag,menunum
 130 str menudat(500),menujob(20),readmenu,crlf,retbat,jobdat
 140 crlf=chr$(13)+chr$(10) : retbat="USEREXEC.BAT"+crlf
 150 width 96 : screen 2,0,1,1 :console 0,31,0
 160 cls :wipe() : color 3
 170 menu_read()
 180 title()
 190 menu_display()
 200 menu_sentaku()
 210 menu_write()
 220 exit()
 230 /*
 240 /*          メニュー定義ファイルの読み出し
 250 /*
 260 func menu_read()
 270 finput=fopen("USERMENU.DAT","r")
 280 mdat=0:po=0:flag=0
 290 while feof(finput)<>-1
 300   freads(readmenu,finput)
 310   if readmenu="$" then {
 320           po=po+1 : flag=1
 330           } else {
 340           if flag=1 then flag=0 : position(po)=mdat
 350           menudat(mdat)=readmenu : mdat=mdat+1
 360           }
 370 endwhile
 380 fclose(finput) : po=po-1
 390 menujob(0)=menudat(0)
 400 for loop=1 to po
 410   menujob(loop)=menudat(position(loop))
 420 next
 430 endfunc
 440 /*
 450 /*          選択メニューの書き出し
 460 /*
 470 func menu_write()
 480 int hajime,owari
 490 hajime=position(menunum)+1
 500 if menunum=po then owari=mdat else owari=position(menunum+
1)-1
 510 foutput=fopen("USERMENU.BAT","c")
 520 for loop=hajime to owari
 530   jobdat=menudat(loop)+crlf
 540   fwrites(jobdat,foutput)
 550 next
 560 if menunum<>0 then {
 570   fwrites(retbat,foutput)
 580   }
 590 fclose(foutput)
 600 endfunc
 610 /*
 620 /*          タイトル表示
 630 /*
 640 func title()
 650 str title="処理選択システムメニュー"
 660 fill(0,0,765,510, 6)
 670 fill(168,56,616,488,0)
 680 fill(160,48,608,480,3)
 690 fill(272,56,480,88,5)
 700 locate 80,2 : print "日付 "
 710 locate 80,3 : print "時間 "
 720 locate 47-int(len(title)/2),4
 730 print title
 740 endfunc
 750 /*
 760 /*         メニューの表示
 770 /*
 780 func menu_display()
 790 str menunum
 800 for loop=0 to po
 810   menunum=itoa(loop)
 820   if len(menunum)=1 then menunum=" "+menunum
 830   locate 25,loop+6
 840   print menunum;" ... ";menujob(loop)
 850 next
 860 endfunc
 870 /*
 880 /*          メニューの選択
 890 /*
```

ムを移植したものなのだそうですが，元になったプログラムがよっぽど使い込まれていたのか，とても使いやすいです。メニュー定義ファイルも書きやすいし，なんともグッドグッド，ですよね。

そうそう，本プログラムの改造，頒布は自由に行ってかまいませんが，プログラムの最初の部分に記述してある開発者などの記述は削除しないでくださいとのことです(ま，当然のことではありますよね)。礼儀を失しないように気をつけながらバシバシ配ってしまいましょうね。

さあて，今月はここまで。今月はX-BASICのプログラムが集中してしまいましたけど，もちろんCだってバッチだって大歓迎ですよ～。求む「Here Comes a New Challenger」チャラッチャッチャラ☆

また来月っ。

```
 900 func menu_sentaku()
 910 int ans=0
 920 str menukey,menunumber
 930 beep
 940 repeat
 950   color 7 : locate 30,28
 960   print "メニュー番号を選択して下さい。"
 970   menunumber=""
 980   repeat
 990         locate 85,2 : print date$
1000         locate 85,3 : print time$
1010         menukey=inkey$(0)
1020         if menukey<>chr$(13) then menunumber=menunumber+
menukey
1030         locate 62,28 : print menunumber
1040   until (menukey=chr$(13) and len(menunumber)>=1)
1050   menunum=atoi(menunumber)
1060   if menunum>=0 and menunum<=po then {
1070         ans=1
1080         } else {
1090         ans=0
1100         locate 30,28 : print space$(60)
1110         for loop=1 to 5
1120         beep
1130         color 5 : locate 30,28
1140         print "その番号は定義されていません。"
1150         for null=1 to 200 :next
1160         next
1170         }
1180 until ans=1
1190 endfunc
```

ぱーてぃハンズ(8)

ええ～，今月のハンズは短いぞ。もうしわけない(完全に隔月体制になれてしまったこの体……，でもなんとか書いたんだから許してっ)。

さて，なんといってもゲームを遊ぶときに臨場感を出してくれるのが，効果音とゲームミュージック。ミュージックのほうは置いておくとして，効果音の鳴らし方をちょっと予習しておきましょう。

X68000で鳴らせる音

えーっと，効果音とひと口に申しましても，X68000には音を出すための道具がいろいろとございます。まずは，FM音源。こいつは昔風にいうところのシンセサイザでございまして，音を人工的に合成して作り出して鳴らすんですね。まあ，人工的に作り出すといってもX68000の場合はプリセットされている音がたくさんあるので，わざわざ自分で作る必要はそれほどないでしょう。

でもってFM音源を陰とすれば陽。X68000の北斗と南斗といわれるのがAD PCM(おい)。こいつはマイクなどからどこかにあった音をメモリ上に取り込んでおいて，必要なときにはそれをそのまま再生するという，九官鳥のキューちゃん(どうでもいいけど九官鳥ってどうして，どいつもこいつもキューちゃんなんだろう？　べつに八っちゃんでもいいじゃないかと思うのだが。オリジナリティって大事だよね)のようなものなのですね。

あとはオプションでMIDIなんていうのがありますね。これはMIDIっていう音の鳴らし方ではなくて，MIDI規格っていう楽器をつなぐ規格があるんです。それ用のインタフェイスカードがX68000ではオプションで使えるので，そこにつなげられる音源装置をMIDI音源なんていうわけです。

で，その機械としてはローランドのSC-55やMT-32なんてのが有名ですね。えーっと，あれはLA音源っていう方法になるんでしたっけ？　ややっこしい。ま，そういうこってす。

で，あとX68000で鳴らせる音っていうと，ファンの風切り音とかハードディスクのアクセスコリコリ音とかがありますが(でもゲームの効果音にハードディスクのコリコリ音が使われてたら心臓に悪かろうな……)，ま，一般的に効果音などに使うことはないだろうから，とりあえず，X68000の音源はFM音源とAD PCMと思っておけば間違いないです。

で，効果音なんですが，このFM音源とPCM音源のどちらを使うか。はっきりいって「どっちでもいい」んですけれども，今回はとりあえず，AD PCMというセンでいきましょう。

効果音を出そう

で，AD PCMの音の鳴らし方なんですが，X-BASICの場合は非常に簡単です。一応，仕組みを説明しておきましょう。

まず，AD PCMではメモリ上にAD PCMデータ，つまり取ってきた音をメモリ上に置ける形にしたもの，を置かなければなりません。AD PCMデータってのはあれですね，よく.PCMってファイルになっているやつ。Z-MUSICのディスクや電脳倶楽部なんかにもたくさん入ってますよね。

で，これをやるのは簡単。まず，dim文でデータを置くメモリを作ってあげます。

dim char PcmMem(10000)

こんな感じですね。そして，ディスクからAD PCMデータをメモリ上に置きます。ディスクからファイルを読む常套手段，fread()関数を使うと以下のような感じになります(PCMデータファイルの名前が"DRUM.PCM"だとする)。

fp=fopen("DRUM.PCM","R")
fread(PcmMem,10000,fp)
fclose(fp)

これでメモリ上にPCMデータをセットすることができました。さあ，あとは鳴らすだけですね。PCM音源を鳴らす命令はと……a_play()があります。ってことは鳴らしたいところでこう書けばいいのですね。

a_play(PcmMem,4,3)

マニュアルによるとa_play()関数は，

a_play(na,sf,md, [lng])

na……PCMデータを格納している数値型1次元配列名
sf……サンプリング周波数
0……3.9kHz
1……5.2kHz
2……7.8kHz
3……10.4kHz
4……15.6kHz
md……出力モード
0……出力カット
1……左出力
2……右出力
3……ステレオ出力
lng……再生する配列naの添字0からの長さ。省略するとすべてのAD PCMデータを再生します。
返り値……なし

だそうですから，この場合はDRUM.PCMの中身を15.6kHz，ステレオでファイルの内容全部を再生するわけです。

ほかにもAD PCMデータの出力の方法としては，Z-MUSICとMUSICZ.FUNCを使ってm_pcmset()とm_trk()&m_play()で鳴らす，なんて方法もありますけどね。

さて，これでサンプリング効果音の出し方がわかってしまいましたね。自分のゲームで「ラウンドワン，ファイト！」でも「ムーンクリスタルパワー(以下略)」でも好きなようにしゃべらせてしまいましょう～(でも，著作権には気をつけてね)。

RED ZONE&5インチFDD

Kioi Makoto 紀尾井 誠

パソコンショップ満開はRED ZONEに対応した5インチ増設フロッピーディスクドライブを発売する。ここでは3.5インチ機のディスク環境を考えてみよう。

●

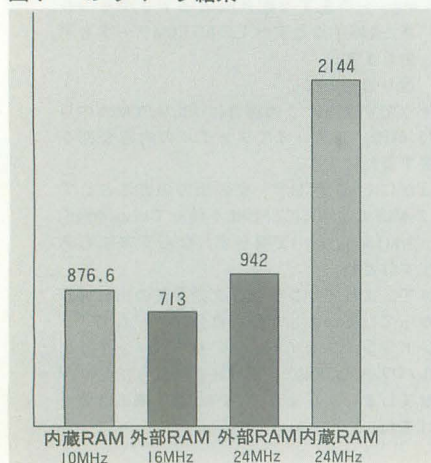
アンケートはがきを見ているとモニタプレゼントのRED ZONEが大人気なようだ。世の中，486の33MHzではちょっと遅いかなといった風潮なのに，68000の24MHzで速いと喜ばれるのを不思議に思う人もいるかもしれない。ソフトウェアの作り方とユーザーの使い方の違いということになるのだが，ハードウェアの違いやアプリケーション，OSの速度など，結局，それは文化の違いということになる。

問題はメモリ？

RED ZONE自体のコストパフォーマンスはそこそこいいのだが，その速度を無駄にしないように本格的に使おうとすると，なにはともあれメモリを増設しなければ話にならない。残念なことにX68000CompactXVIの本体内蔵用のメモリボードはかなり高価である。

X68030用のメモリが純正でさえほぼ半額，IOデータ機器のものではさらにその半額という価格だ。世間はようやく適正な価格になってきたのに，X68000のRAMは相変わらず高い。

こういったことからRED ZONEとX68030 Compactを実売価格帯で考えると，12Mバイトまで増設したときには価格はさほど変わらなくなってしまうのだ。

図1　ベンチマーク結果

5万円出せばフル増設になるX68030に対し，2Mバイトしか増えないX68000ではちょっと分が悪い。快適な環境を構成するための標準的なメモリ容量を6Mバイトだとすると，10万円の投資が必要になる。しかし，買い換えや買い足しで導入する場合にはすでに拡張スロット用のメモリボードがある人も多いだろう。拡張スロット用のRAMなら，かなり割安で増設できる。

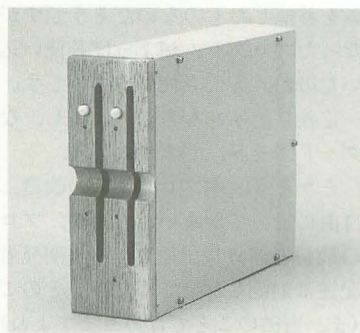
ここで内蔵RAMを使用しない場合のパフォーマンスを調べてみよう。ここでは拡張スロットに4Mバイトの増設メモリを搭載してみた。

まず，拡張スロット上のRAM内だけで動作するようにデータとプログラムを置く。先月と同じDhrystone(ver.2.0, XC ver.2.1でコンパイルしたもの）のベンチマーク結果は図1のとおり。

やはり内蔵用のRAMを使わないと性能は発揮できない。24MHz時で11MHz相当，16MHz時では8MHz相当にまで落ち込んでしまう。

ちなみに，今回使用したのは手持ちのIOデータ製で，RAMには80nsのものが使用されている。編集室にあった純正のCZ-6BE4を見るとRAMには120nsのものが使用されており，それを使った場合，24MHz時で737.1，16MHz時で621.5，10MHz時725.0dhrystoneという結果が出た。

こういったまちまちの値が出てくるのは，メモリアクセスのタイミングによって，たまに1ウエイト入ったり，2ウエイトだったり……ということが起こるからだ。これはシステムクロックにかかわらず拡張スロットが10MHz動作をしていることによる。16MHzで1ウエイト入るよりも10MHzでノーウエイトのほうが速い，といったこともある。タイミング次第だから，使用しているRAMボードによってかなり差が出てくることが考えられる。同じ製品でも発売時期によってRAMの種類にばらつきがあるはずなので，ベンチマーク結果はあくまで参考程度に留めておいてほしい。

ベンチマーク結果からは，本体内のメモリは24MHz時でもノーウエイト動作をしていることがうかがえる。性能をフルに発揮させるためには内蔵用を選択すべきだ。内蔵RAMが4Mバイトでさらにスロット

満開版5インチFDD（の模型）

にいくらかのRAMがつけば，これはもう文句のつけようがない。遅いRAMといってもRAMディスクでの使用ならばさほど気にならないのだ。

さらに，実使用感として拡張スロットのRAMを使っていると遅いか？　というとそうでもない。

もっともメモリを消費しそうなSX-WINDOWでの動作ではウィンドウの開閉がやや重いかなという感じも受けるが，メモリが足りないときの状況に比べればはるかに使える環境といえるだろう。コマンドラインから実行されるようなたいていのプログラムは先頭2Mバイトの高速な部分をもアクセスするので，実際には先ほどのテストほどの速度差は感じられない。メモリロード時に実行ファイルは前方に，そのプログラムで使うデータエリアがその後ろに確保されることから，プログラム自体は高速だけどデータアクセスで遅くなるといった感じだろう。

結論としては，「拡張スロットのRAMでも十分使いものになる」という感触を得た。余裕ができたら内蔵用を2Mバイト加える，これで十分だろう。

増設5インチFDドライブ

パソコンショップ満開ではX68000CompactやRED ZONEをどうやら「入門者用に」販売しているらしい。RED ZONEならば価格的にも性能的にも問題はないだろう。しかし，X68000ではこれまでのソフトウェア資産のほとんどが5インチFD上のものである。X68000Compactだけでは市販ソフトなども使えるものが限られてしまうので，やはり5インチのフロッピーディスクドライブは揃えたいところだ。

ということで，パソコンショップ満開では，5インチの増設フロッピーディスクドライブを販売することになったらしい。もちろん2ドライブ構成で価格は39,800円(税別)。これはパソコンショップ満開のオリジナル製品で，なんとオートイジェクト

のメカを搭載したきわめてまっとうな製品である。純正品と同じドライブを使用しているので互換性にはまったく問題がない。もちろん，X68000Compactだけでなく，X68030Compactでも使用可能だ。

フロントパネルは木製で本体部分とフロントパネルのカラー塗装の具合で3タイプ，すなわち，

本体未塗装，フロント未塗装
本体塗装，フロント未塗装
本体塗装，フロント塗装

の3種類が用意されている。さらに，特注でフロントパネル部分に任意の木製材質を指定できるというサービスも行われるという。本体のチタンブラックにマッチする黒檀，紫檀などの高級素材も面白いだろう。木製家具を使用しているなら，チークやマホガニー，ローズウッドといった化粧合板の飾りにされているような素材が似合うかもしれない。

3.5インチFDドライブとして

とはいえ，新規ユーザーならともかく，買い足してRED ZONEやX68030Compactを導入する人というのも少なくないだろう。ここで考えるのが，これまでの5インチFDD搭載機を増設ドライブとして使用できないかということだろう。

これまでもいろいろ憶測はあったのだが，結論として，これは可能である。先月のCGA講座でもちょっと触れられていたように，FDケーブルで5インチ搭載機と直接つなぐのだ。これにより5インチ搭載機はCompactの増設5インチFDドライブとして（ただしドライブ番号は0，1に固定），Compactの内蔵ドライブは従来機の増設3.5インチFDドライブとして使用できる，というのだ。

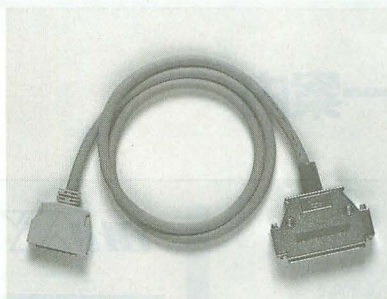
結構ありそうなパターンとして，5インチ従来機と拡張メモリ4Mバイト，SCSIハードディスク（相当品）がすでにある場合にRED ZONEを導入することを考えよう。

写真は都内某所にある，とある編集者の机の上のシステムだ。接続構成図は図2のようになる。接続用のケーブルは秋葉原のツクモ電機で完成品を販売しているので，それを使ってみた（9,800円税別）。

SCSI直結ではアービトレーションの問題があるのだが，ひとりで使っている分にはディスクアクセスがぶつかってハングアップする危険は少ないだろう。

フロッピーディスクの場合はどうだろうか，7月号のDōGAの連載中の囲み記事には「確実にディスクを破壊する」とあった。

このケーブルで直結する

接続例。かなり使えるシステムだ

片方がフロッピーにアクセス中にもう片方から違うドライブにアクセスしてみる。すると同時にアクセスした場合はあとからアクセスしたほうに「無効なメディアを使用しました」エラーが発生するが，片方のアクセスが終わるのを待ってリトライすれば大丈夫なようだ。かなり危なっかしいが，多分ディスクを破壊することはないだろう。まあ，いずれにせよ信号レベルで見るとかなり気持ち悪いことをしているので避けるにこしたことはない。

物理的なディスク破壊以外にも，片方でいじっているときにファイルの内容をいじられたり，新規ファイルを作成されたりするとちょっとマズイことが起こると考えられる。メモリ上に持っているディスクの管理情報と実際のものが食い違ってくるのだ。これにより，FATやファイルが破壊されることが考えられる。

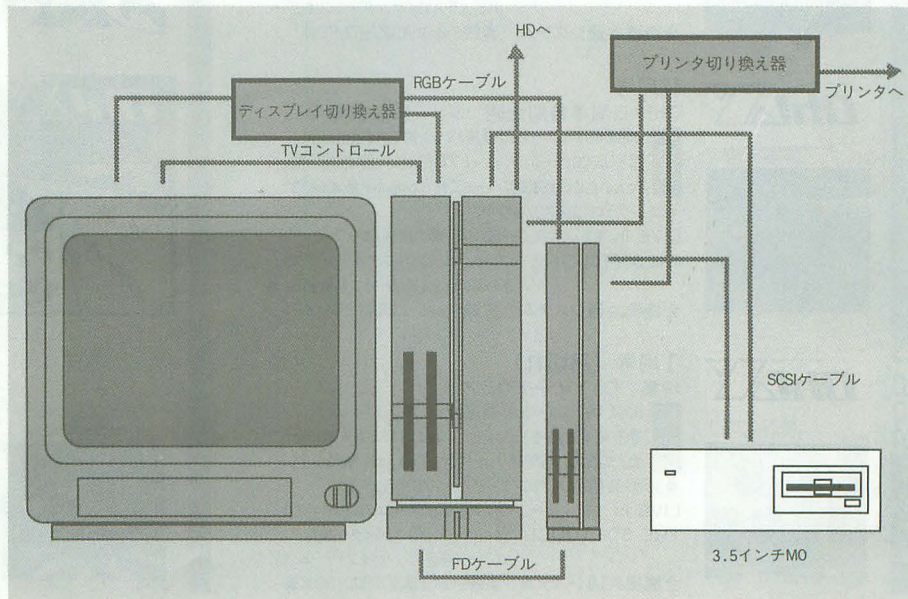
対策としては，メモリ上にあるディスクの管理情報はブレイクキーを押せば初期化されるのでディスク操作の前に必ずブレイクキーを押すようにすればよい。

写真にあるシステムでは，CONFIG.SYSで設定する“buffers”の値を2にしておく，という方法で危険を最小限に抑えている。

なんであれ，ファイルアクセスの際には必ずFATとディレクトリの部分を必要とするので，最低でも2つのバッファを参照すると考えていい。よって，なにかひとつのファイルにアクセスするとそれ以前の情報は綺麗に消えてしまうのだ（ただしSX-WINDOWは別）。

さて，“buffers＝2”とすることによる弊害としては，ディスクアクセスが頻繁に発生するようになるということがある。たっぷりバッファを取っておいた場合に比べて読み込みが遅くなる（ディスクアクセスが多くなる）。もちろんFASTIOなどのディスクキャッシュなども使えない（ディスクキャッシュはドライブごとに設定できるべきだよな，やっぱ）。

これでLANに対応したドライバがあれば，アクセスのぶつかりなどを気にせず，さらには直接には接続されていないハードディスクやRAMディスクを外からアクセスできるようにもなるだろう。ディスクバッファを最小にしたりしなくてもよくなるはずだ。このあたりは先月から連載されている中井氏の記事に期待しよう。

図2

バックナンバー案内

BACK ISSUES

ここには1992年8月号から1993年7月号までをご紹介しました。現在1992年6，7，9，12，1993年4～7月号の在庫がございます。バックナンバーはお近くの書店にご注文ください。定期購読の申し込み方法は156ページを参照してください。

1992

8月号（品切れ）
特集　プログラミング再入門
連載　響子 in CGわ～るど/吾輩はX68000である/よいこのSX-WINDOW/マシン語プログラミング/ハード工作/ANOTHER CG WORLD/大人のためのX68000/Computer Music入門/ショートプロ
●新製品紹介　MATIER/TG100/SOUND SX-68K
LIVE in '92　氷穴/ガラガラヘビがやってくる/風の贈り物
THE SOFTOUCH　三國志III/シムアース/ウルティマVI/バトルテック
全機種共通システム　実践Small-C講座(5)ワイルドカード　グラフィックライブラリGRAPH.LIB

9月号
特集　数値演算の熱い逆襲
連載　DōGA CGアニメーション講座/大人のためのX68000/響子 in CGわ～るど/吾輩はX68000である/ショートプロ/マシン語プログラミング/ハード工作/ANOTHER CG WORLD
●新製品紹介　MATIER/MIREGE Model Stuff
LIVE in '92　恋をしようよ Yeah! Yeah!/ゆめいっぱい
THE SOFTOUCH　ファイナルファイト/ライジングサン/ヨーロッパ戦線/シューティング68K GAMES
全機種共通システム　O-EDIT & MODCNV

10月号（品切れ）
特集　DTMへの招待
連載　DōGA CGアニメーション講座/大人のためのX68000/響子 in CGわ～るど/吾輩はX68000である/ショートプロ/マシン語プログラミング/ハード工作/ANOTHER CG WORLD
●試用レポート　X68000用CD-ROMドライブ
LIVE in '92　美少女戦士セーラームーン/笑顔を探して 他
THE SOFTOUCH　ポピュラスII/リーディングカンパニー/ネクタリス/サークII
全機種共通システム　実践Small-C講座(6)SLENDER HUL

11月号（品切れ）
特集　ゲームマネージメント
連載　DōGA CGアニメーション講座/大人のためのX68000/響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW/ハード工作/ANOTHER CG WORLD/Computer Music入門
●新製品紹介　CHART PRO-68K
LIVE in '92　ストリートファイターII/スーパーマリオ 他
THE SOFTOUCH　キャッスルズ/シュートレンジ/ポピュラスII/サンダーレスキュー
全機種共通システム　実践Small-C講座(7)EDIT

12月号
Oh!X 5周年特別企画　ショートプロ大集合
連載　DōGA CGアニメーション講座/マシン語プログラミング/響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW/大人のためのX68000/ハード工作/Computer Music入門
●エレクトロニクスショウ'92
LIVE in '92　LAST CHRISTMAS/闇の血族/ユーフォリー
THE SOFTOUCH　デスブレイド/ムーンクレスタ&テラクレスタ/ふしぎの海のナディア/ロードス島戦記II 他
全機種共通システム　実践Small-C講座(8)MAKE

1993

1月号（品切れ）
特集　D.I.Y.ハードウェア
連載　DōGA CGアニメーション講座/マシン語プログラミング/響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW/大人のためのX68000/ハード工作/Computer Music入門
●新製品紹介　サンダーワード/SX広辞苑
LIVE in '93　ムーンライト伝説/チャコの海岸物語
THE SOFTOUCH　オーバーテイク/ストライダー飛竜/エアーマネジメント/パイプドリーム 他
全機種共通システム　実践Small-C講座(9)EDC-Tの拡張

2月号（品切れ）
特集　画像創造のために
連載　DōGA CGアニメーション講座/マシン語プログラミング/響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW/ハード工作/吾輩はX68000である/Computer Music入門
●新製品紹介　Communication SX-68K
LIVE in '93　FIRE CRACKER/サンバDEグワッシャ!
THE SOFTOUCH　極/ドラゴンスレイヤー英雄伝説/機甲装神ヴァルカイザー/キングス・ダンジョン
全機種共通システム　BLACK JACK

3月号（品切れ）
特集　X-BASICを学ぶ
連載　DōGA CGアニメーション講座/マシン語プログラミング/響子 in CGわ～るど/ANOTHER CG WORLD/ハード工作/ショートプロ/Computer Music入門/Z80's Bar
●緊急速報　32ビットマシンX68030
●新製品紹介　音源モジュールSC-33/GS音源搭載JW-50
LIVE in '93　ストリートファイターII/晴れたらいいね 他
THE SOFTOUCH　究極タイガー/チェルノブ/シムアント 他
全機種共通システム　シューティングゲームコアシステム作成法(1)

4月号
特集　X68第7世代へ
連載　DōGA CGアニメーション講座/マシン語プログラミング/響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW/ハード工作/吾輩はX68000である/Computer Music入門
●決定! 1992年GAME OF THE YEAR
●名作ゲーム再遊記
LIVE in '93　FIGHTMAN/ミンキーモモより 愛しのマーシカ
THE SOFTOUCH　スターフォース/元朝秘史 他
全機種共通システム　シューティングゲームコアシステム作成法(2)

5月号
特集　襲撃! SX-WINDOW
第8回　言わせてくれなくちゃだワ
連載　DōGA CGアニメーション講座/ANOTHER CG WORLD/響子 in CGわ～るど/ショートプロ/大人のためのX68000/ハード工作/吾輩はX68000である/Computer Music入門
●X68030へのソフトウェア対応について
LIVE in '93　MAGICAL SOUND SHOWER/もう笑うしかない 他
THE SOFTOUCH　エトワールプリンセス/メガロマニア 他
全機種共通システム　シューティングゲームコアシステム作成法(3)

6月号
創刊11周年特別企画　確率遊技シミュレーション
連載　DōGA CGアニメーション講座/こちらシステムX探偵事務所/響子 in CGわ～るど/ショートプロ/大人のためのX68000/ハード工作/吾輩はX68000である/Computer Music入門
●新製品紹介　SC-55mkII
LIVE in '93　ストリートファイターIIより 春麗のテーマ/BAY YARD/LOVE&CHAIN
THE SOFTOUCH　餓狼伝説/信長の野望・覇王伝 他
全機種共通システム　REVERSI

7月号
特集　席巻するローテク文明
連載　DōGA CGアニメーション講座/こちらシステムX探偵事務所/響子 in CGわ～るど/ショートプロ/マシン語プログラミング/ハード工作/吾輩はX68000である/Computer Music入門
●新製品紹介　ドローイングバット33070&MATIER
LIVE in '93　Midnight Circle/今日の日はさようなら/赤い靴
THE SOFTOUCH　悪魔城ドラキュラ/リブルラブル/大航海時代II/銀河英雄伝説III/幻影都市/ヴェルスナーグ戦乱
全機種共通システム　MSX用S-OS "SWORD"

こちらシステムX探偵事務所

来月はモーフィング実験だ！

Shibata Atsushi 柴田　淳

「こちらシステムX探偵事務所」も3回目になり，順調にその仕事をこなしているようですね。それにしても，回を重ねるごとに柴田氏のいうことがドンドンでかくなっているような気がするけど，大丈夫かなあ。

FILE-Ⅲ

illustration: T. Takahashi

マスター（以下M）：お待たせしました。次の方どうぞ。

電話番候補の女の子その2（以下候2）：ヤッダー，この部屋どうしてこんなに暑いのー。お化粧くずれちゃうー。

琴張護（ことわりまもる）**（以下護）**：ええと，さっそくですがお名前から。

候2：わたし晶紀ちゃん。

M：どうしてこんなのばっかり……。

護：なるほど名前といわれて名字を答えないとは，なかなか論理をわきまえていらっしゃる。

柴田淳（以下Ats）：そういう問題じゃないような気がするけどなあ。

M：ところで，今回応募なさったきっかけを聞かせてもらえますか。

候2：ええとー，探偵事務所っていうとカッコよさそうっていうかー，なんだか危険でいっぱいかなー，みたいなー。

護：格好だとか危険だとかはさておき，今回は電話番を兼ねた事務員のようなことをしてくれる人を募集したのですけど。

候2：電話番ならばっちりー，みたいな。晶紀ちゃん長電話大好きー，みたいな。

M：事務所の電話を私用に使われると困りますよ。

Ats：それじゃあ事務の経験はあるんですか。

候2：小学生のころー，ソロバンとお習字やってたしー。

護：ところで先ほどから気になっていたのですが，その体中につけておいでの貴金属はどうなされたのですか。

候2：あ，これー？　これねー，晶紀ちゃんの彼に買ってもらったのー。彼ってすごくお金持ちでー，車はBMだしー。

Ats：そんなこと聞いてるんじゃ……。

候2：仕事はキャッチセールスやっててー，月収150万円でー，バリバリのエリートって感じー。晶紀ちゃん彼の仕事中にナンパされてー，ついでに100万円の英会話ビデオも買っちゃったー。

M：ハイハイそれはようございましたね。

護：ところでその貴金属のことですが，私が思うにあなたのような女性がつけるよりも工業利用したほうが，よほど世の中のためになるんじゃないかと。この大きなダイヤモンドもですね，粉々に砕いて研磨材にしたほうが……。

候2：やだー，なにすんのよー，みたいな。もうこの人さいてー。晶紀ちゃんもう帰るー，みたいな。

（ガチャガチャバタン）

M：はあ，前回応募のあった人をとりあえず面接に呼んでみたのはいいものの，やっぱりいい人はいませんでしたね。

護：あのような女性を雇うのは，給料をドブに捨てるようなものです。

Ats：まったくですよ。だいたいキャッチセールスなんて，人をだまして稼いだような金で買った貴金属を，これみよがしにつけてる人間の気がしれないですよね。

M：それにしても，電話番どうしましょうか。職業柄，留守番電話で注文を，ってわけにはいかないしなあ。

護：実はそのことで提案があるのですが。

Ats：どうしたんです，急にかしこまっちゃって。

護：実は私の知り合いに電話番にうってつけの人物がいるのですが，なにぶんその人物というのが身内なのです。だから話したものかどうかと……。

M：琴張さんがいやだというならしょうがないけど，このさい身内でもなんでもかまいませんよ。状況は切迫してるんです。

護：それは私も知っています。ですからあくまでも最後の手段ということでして。

Ats：なにもったいつけてるんですか。誰なんです，妹さんですか？

護：ま，まあそのようなものですけど，それなら連絡を取って，面接に来るように伝えます。

座標変換の極意

Ats：どうしたんでしょうねえ琴張さん。いつも無鉄砲に強気の人なのに。

M：で，その琴張さんから話は聞いていると思いますけど，例のやつできてますか？

Ats：ええもちろん。だいたい今日ここに来たのだって，その仕事を届けるのが目的だったんです。グラフィック画面上で，三角形どうしの自由変形をするサブルーチンということでしたよね。

M：いつもすいませんねえ。そういえばこの仕事，前回テキストの三角形描画ルーチンを頼んだ同じ人からの依頼なんですよ。

Ats：なるほどね。三角形を塗り潰せるようになると，その次はそれに画像を張りつけたくなるってわけですね。この連載も脈絡がないようでいてその実，段階的な内容発展を見せているところがすごいよな。

護：自画自賛は人格退行の前兆です。あなた気をつけないと将来ボケますよ。

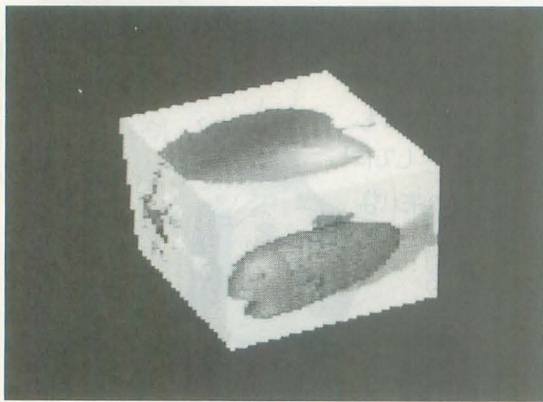
Ats：あっ，琴張さんいつの間に。しかも高飛車な性格が戻ってる。

M：ところで，過去2回の内容に比べて，今回の三角形どうしの自由変形ってずいぶん難しそうじゃないですか。

Ats：そんなことはないですよ。前回，前々回の内容がわかっていれば，少なくともどのようにして変形を行っているかを理解することだけは，できるはずです。

M：でも，片方の画像をそのまま別のところに張りつけるのならまだしも，形を変えてとなると難しそうに見えますよ。

Ats：今回は，ペイント系のグラフィックエディタなんかによくついている，パースペクティブの方法に似ているんです。

M：パースペクティブって？

これがマッピングだ

護：主に長方形内の画像を，任意の四角形内に張りつける機能のことです。奥行をつけたような結果が得られるので，このように呼ばれています。

Ats：たとえば，正方形の中の絵を，上の辺をつぼめた台形の中に張りつけるときを考えましょうか。すると，正面を向いていた板が，上を向いたような感じに変換されますよね。

M：奥まった辺は幅がせまくなりますからね。

Ats：じゃあ，いまのような変換を台形の底辺から上に向かって，画像を一段ずつ張りつけていくことで実現するとします。いちばん下の辺では，変形する元の画像と変形する先の画像は変化しないはずですよね。

護：確か底辺の長さは，元と先では同じでした。

Ats：でも積み上げていくにしたがって，変形先の横幅は狭くなっていく。すると元の画像をそのまま変形先に書き出すのでは都合が悪いから，画像の情報を間引かなくちゃならない。

M：そうですよね。しかも端や真ん中で集中的に間引くと，そこだけ元の内容がごっそり抜けてしまうから，間引き方にバラツキをもたせなければならないんだろうな。

Ats：またそれとは逆に，元の画像より幅の広い部分に向かって変形を行うときには，今度は画像情報を引き伸ばさなければならないですよね。

護：すると間引くにしても引き伸ばすにしても，それに適当なバラツキをもたせるにはどうすればいいかというのが第一の問題のような気がしますが。

Ats：そうなんですよ，その「適当なバラツキ」をどうするか，という問題を解決するのが，座標変換という考え方です。端的にいえば「変形元の図形の中の1点の座標が変形先のどこに当たるのか」を教えてくれる，変換式を用意すればいいんです。

M：もうちょっとわかりやすく話してくださいよ。

Ats：そうですねえ，じゃあ2人の人間，AとBが，次のような方法で同じ場所の地図を作るとしましょう。

・出発地点から東に向かって歩いていく。

・1分ごとに立ち止まり，周りにあるものを紙に書いていく。

・一定距離進んだら元の位置まで戻って，ある程度北上し，また東に歩き出す。

護：同じ場所でという条件がありますから，この方法では2人とも同じ地図ができあがります。

Ats：いや，そうともかぎらないですよ。たとえばAのほうがBより，歩く速さが2倍だったらどうですか。

M：そうか，周りにあるものを書き込む間隔は1分なんだから，Aのほうが2倍の距離進むわけだ。

Ats：この場合，2人の地図を比べるとどうなるでしょう。

護：Aの地図の幅がBの地図の2分の1になります。

Ats：じゃあこんなのはどうですか？ Bは実はすごくスタミナのない奴で，西に戻って北上するたび歩く速度が遅くなっていくとしたら。

M：ええと，歩く速度が遅くなっていくということは地図の幅が広がるということだから。

護：北方向に広がっていく，逆さの台形のような形をした地図ができあがります。

M：と，いうことはですよ，できあがる地図の形を目的の台形にしようと思ったら，東に歩く速度を調節すればいいってことになりますよね。

Ats：その歩く速度をどのくらいに調節すればいいかを決めるには，変形元の四角形の幅と，変形先の幅を比べるんですけど，すると先ほど話した「座標変換のための式」が求まるんです。これが，長方形から台形に画像を変形させるための，大まかな方法かな。

正方形を変形する

護：さていまの方法で，長方形から台形への変形は実現できます。しかしグラフィックエディタのパースペクティブでは，変形先がどんな形であれちゃんと変形してくれるはずですが。

Ats：そうなんですよね。この方法は，あくまでも画像変形に向けてのとっかかりというか，出発点でしかないんです。ではもう少し突っ込んだ話をするため，先ほどの地図を描く方法を，以下の新しい条件に変えてみましょう。

・Aは一定速度で，1キロ四方の正方形の中を，さっきと同じように進むとする。

・一定距離進んだら，正方形内の縦横の比率で，自分のいる位置をBに伝え，ついでに周りにあるものも伝える。

護：正方形内の縦横の比率とは，出発点から見て，1キロのうちどのくらい進んだかということですね。

Ats：で，その比率を受け取ったBは，正方形でない「歪んだ」四角形の中に地図の情報を書き出すんですが，どこに書き出すかというと，

・Aから伝えられた比率から，対応する辺ごとに同じ比率の内分点を割り出し，その内分点を縦横に結んだ2つの線の交点の位置に情報を書き出す。

護：ずいぶんややこしい表現ですね。

Ats：こういうのは文章で説明するより図で表したほうがわかりやすいので，図1を見てください。

M：なるほど。いってみれば，南京玉すだれにマジックで点を打って，それを変形させたあとの点に地図情報を書いていく要領ですね。

Ats：あとはこの操作を繰り返すことで，正方形を歪んだ四角形に変形できます。

護：いや，でも待ってください。この方法だと不都合があります。変形元の正方形領域と比べて，変形先の四角形のほうが非常に大きかった場合，変形先に書き込む情報がまばらになってしまいます。

M：そうか，Aは自分の位置情報を比率で伝えるから，Aが一定距離動いているつもりでもBが大きく動くことになるのか。

Ats：それを避けるためには，さっきの地図作成の手順の順番を多少入れ替えればいいんですよ。こんなふうにね。

・Bは変形先の歪んだ四角形の中を，隙間なく移動する。

・移動するたび，先ほどAから伝えられた比率を使って自分の位置を割り出すのとは逆の方法で，自分の位置を比率で表してAに伝える。

・AはBからの比率どおりの場所に移動し，Bにその場の周りの情報を伝える。

・Bは自分のいる位置に情報を書き込む。

M：Aが位置を伝えるんじゃなくて，Bが伝えるようにすればいいのか。

Ats：南京玉すだれのたとえでいくと，まずすだれを変形させてからマジックで点を打って，次に元の形に戻すというふうになりますかね。
護：変形先を隙間なく移動するのだから，確かに書き込む情報がまばらになることはなさそうです。
Ats：で，この方法はまっとうに考えたものと逆の手順を踏むから「逆変換」なんて呼ばれたりします。

三角形ではどうか

M：ところで，この方法だと長方形から任意の四角形に変形はできるけど，歪んだ四角形どうしの変形はできませんよね。
護：そういえば，グラフィックエディタのパースペクティブでも，変形元の画像は常に長方形です。
Ats：できないことはないはずなんですよ。だってさっきの地図のたとえでいうと，BがAに位置を指示したあと，AがBから教わった比率をもとに，Aのいる歪んだ四角形内での場所を割り出せばいいんですからね。
護：つまり逆変換を変換すればいいのですね。
Ats：ちょうどいいところで「変換の変換」という話が出てきたんで，これから一気に三角形どうしの変形の話に入っていきましょう。
M：ということは，変換の変換をすれば三角形の自由変形ができるってことですか。
Ats：まあ，そう先を急がずに。まず，先ほどの四角形の自由変形と三角形の場合を比較することから始めましょう。四角形の場合，辺は4つあるんだから，比率から座標を求めるのは比較的簡単ですよね。
護：底辺と上辺，左右の辺がはっきりしていますから。
M：ところが三角形の場合はそうはいかないと。
Ats：俗にいう「座標系」をどのように記述するか，という問題なんですけど，3月号の特集にある中野氏の記事では，いちばん上の頂点から底辺に向かって，目的の点を通るような線を下ろして，その線上での点の位置と，底辺と線との交点の比率を使って三角形内の座標を変換してましたよね（図2）。これが「変換の変換」に当たるものなんです。
M：じゃあその方法を使えばいいんじゃないですか。
Ats：だけど今回は，座標の記述に違う方法を取っています。
護：どうしてわざわざ。
Ats：理由はあとで説明しますけど，まあどんな具合にやっているのか聞いてくださいよ。まず，中野氏の方法だと記述に必要な比率は2つですが，今回はひとつの点を表すのに3つの比率を使っています。
M：3つっていうと？
Ats：まずなんでもいいから，目的の点を通るような直線を考えます。その直線は，必ず三角形の2辺と交わるはずですよね。その交点の間の線分上で，目的の点がどの位置にあるのかを第1の比率とします。
護：それではほかの比率はどうなるのでしょう。
Ats：いま引いた線分の，三角形上の辺との2つの交点が，それぞれ交わる辺のうちでどの位置にあるのかが残りの2つの比率なんです（図2）。で，面白いのは，この変換と中野方式で得られる座標は，まったく同じになるんです。
M：どうしてでしょうかねえ。
Ats：詳しい証明はここでは避けますが，まあとりあえず，この変換方式でも問題はないということだけ頭に入れておいてください。
護：同じ結果が出るのに，どうして新しく変換方式を作ったのですか。単なる無駄のような気がしますけど。
M：でも新しい方式をでっち上げるくらいだから，こちらのほうがなにか都合がいいんだろうな。
Ats：そうなんですよ。どうして都合がいいかを説明するために，先月の記事を思い出してもらいたいんですけど。
M：先月の記事というと，塗り潰された三角形を書き出すルーチンですね。
Ats：先月のルーチンでは，上から順番に三角形のふちを探していって，2つのふちの間を塗り潰すという方法をとりましたが，今回の三角形の自由変形でも，それと同じような処理が必要だというのはわかりますよね。
護：変形先の三角形を書き出すとき，この処理が必要なのではないでしょうか。
M：この方法で変形先の座標を求めて，1点ごとに，中野方式で変形元の座標に変形していくとすると，実数を使えるシステムならまだしも，整数しか扱えないマシン語でやるとなると，確かに辛いものがあるかもしれないですね。
Ats：そこで，僕の方法を使うことにしたんです。この場合便利なのは「目的の点を

図1

それぞれ対応する辺の同一比率の点を結び，変換先の点の座標を得る

図2

中野式の座標変換

柴田式の座標変換

のドット数を回数として，均等に移動しながら始点から終点に達する点を次々に生成できればいいわけですよね。

M：つまりそうすれば，変形先の1段を描くごとに，その段に対応する変形元の座標が得られるわけですね。

Ats：ここで，前々回やったラインルーチンを思い出してほしいんです。

M：確かラインルーチンでは，XなりYなり，始点と終点の差が大きい方向をひとつずつ増やしていくことで，線を引いたんでしたよね。

Ats：で，このひとつずつ増やして終点まで，という言葉に実は裏の意味があるんです。

護：それはどういうことですか。

Ats：たとえば縦長の線を引くとすると，ラインルーチンの考え方というのは「縦の差を回数として，その回数分繰り返すことによって，均等に移動しながら終点まで達するように描画する」ともいい変えられるじゃないですか。

M：なるほどね。するとパラメータ計算のときにY方向の差分を代入していた部分に，均等に割りたい回数を入れればお目当ての点が求まるってことですね。

護：しかしこの場合は，一度に数ドット移動することも考えられますから，先月の整数部分の増分と小数部分の増分という概念を導入したほうがよいのではないでしょうか。

Ats：そのとおりです。そのほか，ラインルーチンではX，Yのうち片方は1ずつ増やせばよかったのが，ここでは増減はまちまちだから，パラメータをX軸用とY軸用の2つ用意しなければなりません。

M：これで線分の内分点を求めるという問題は解決しましたけど，ほかになにか問題はないんですか。

Ats：あとは，いままでのことを組み合わせて，僕の方式での座標変換がうまくいくようにすればいいだけです。ちなみにこのルーチンでは，変換先，変換先の「対応する輪郭上の点」をバッファに書き出して，そのあとバッファからその座標を読み出しながら三角形を書いていくという方法を取っていますけどね。

通るような直線」にはスキャンコンバージョンで引く真横の線を当てればいいというところなんです。

M：でも，その直線が三角形の辺と交わる点の比率を求めるのが，結構ややこしそうだけど。

Ats：そんなことないですよ。スキャンコンバージョンでは，横に1本ずつ線を引いて行くんだから，たとえばn回目に横線を引くときは，「交わる線の縦の長さ分のn」の位置に交点があることになるじゃないですか。

ライン描画ルーチンの応用

護：その方法で「三角形の辺と線分が交わる位置の比率」が求まるにしても，その比率から変形元の辺の上の座標を決めるのにはやはり実数演算が必要なのではないでしょうか。

M：そうですよね。確か直線上のm対nの内分点を求める式は，

$$X=\frac{nX_1 \quad + \quad mX_2}{m+n}$$

$$Y=\frac{nY_1 \quad + \quad mY_2}{m+n}$$

とかいうのでしたよねえ。

Ats：ふっふっふ。アマい，アマいなあ。

M：あっ，なんですかその人を小バカにしたような笑いは。

Ats：いいですか，スキャンコンバージョンでは，横線を1列ずつ積み重ねて三角形を描いていくんだから，三角形の縦方向のドット数だけ，その処理を繰り返せば三角形を描き終えられますよね。

護：それはそうですが。

Ats：それなら，変形元の三角形の辺を，縦

M：ところで，変形の原理がわかったところで，どんな具合にできあがったのか見せてくださいよ。またサンプルプログラムを作ってきたんでしょう。

Ats：ええと，一応画像の張りつけられた直方体が回転するというサンプルを作ってきたんですけど。

M：おおっ！

護：これも一見の価値アリ！

Ats：ええと，赤い玉が4つ並んでいるだけじゃちゃんと変形しているのかどうかわかんないだろうから，リストとしては掲載してませんが，このサンプルに少し手を加えたものも用意してます。6種類の画像が，6面に張りつけられているというやつでして。なお読者の皆様には，写真にて堪能していただきたいと。

護：なるほど。きちんと65,536色に対応しているわけですね。

M：あっ，でも待ってくださいよ，ここんとこに変形誤差があるような……。

Ats：ああ，その誤差ですか，予想はしてたんですけどね。

護：わかっていたなら直せばいいのではないですか。

Ats：いや，その誤差っていうのは先月の「太った三角形とやせた三角形」という部分に関係あるんですけど，それほど大きな誤差ではないですし，ソースも長くなるんでそのままにしてあるんです。

M：ところで3月号の特集で，あとは三角形の自由変形ができればみたいなことを書いていたじゃないですか。

護：モーフィングの話ですね。あのあと否定的な見解もあったようですが。

Ats：ただ，せっかくここまで作ったんだから，とりあえずやってみようとは思ってます。

M：とすると，来月あたりはもしかしてもしかすると？

Ats：ただ，この自由変形ルーチンでは，モーフィングに使うには少し物足りない気がします。だいいち誤差を取り除かなければならないし，アンチエリアシングなんていうのもできるようにと，いま考えているところなんです。

護：いいのですか，そのように大きなことをいったりして。

Ats：なるようになるでしょう。

M：本当に大丈夫なのかなあ。

♪カラーン，コローン

Ats：あれ？　誰かきたみたいですよ。仕事の依頼かなあ。

琴張春香（ことわりはるか。以下春）：もう

やっだー，まもるちゃんったら突然呼び出すんだもん。
護：いえ，ですからもしかしたらこういうことになるかもしれないと……。
春：それにしても当日になってからっていうのはないじゃない。わたし今日，生け花教室キャンセルしてきたんだからね。
護：本当に申しわけないです，返す言葉もありません。
M：あの，失礼ですがどちら様でしょうか。
春：ほらあ，突然呼び出しておいて紹介もしてくれないんだから。
M：もしかすると電話番をしてくださるという方じゃないですか？
春：はい。琴張春香と申します。あとこれ，よかったら皆さんで食べてください。
M：いやいや，これはごていねいに。
Ats：名字が同じってことは，やっぱり妹さんでしょう。
護：そうじゃないんですが……。
M：じゃあ，お姉さんですか？
護：妻です。
Ats&M：え？
護：妻なんです。
春：やっだーまもるちゃん，皆さんの前で妻だなんて，恥ずかしいじゃなーい。
Ats：……。世の中狂ってる。
M：……，あんたがだろ。

つづく

三角形の自由変形ルーチンについて

例によって，三角形自由変形ルーチンはCの関数として設計されている。

trtrfm(pre_p,aft_p);

のようにして呼び出す。pre_pとaft_pはそれぞれ変形元と変形先の三角形の3つの頂点を表す構造体のポインタで，

struct TRIANGLE {
int xl,yl,x2,y2,x3,y3 } ;

などと，すべてintで指定する。以上のように呼び出すと，グラフィック画面の任意の三角形領域を，指定の三角形内に変形してくれるが，少々誤差があるのでそのつもりで。

いわずもがな，サンプルプログラムはCで書かれており，コンパイル＆リンクすると，画像が張りつけられたサイコロのようなものがグルグル周り，なんともお得な気分が味わえる。

毎度のことだが，実行中スペースで一時停止，ESCキーでコマンドラインに抜ける。

リスト1

```
  1: /*
  2: ﾎﾞｹ :マッピングされた直方体を回転させる
  3:       名づけて
  4:       「画像の張り付けられた直方体をグルグルまわす」
  5: ﾂｯｺﾐ:ソノマンマやないかい！(*)
  6:
  7: 先月と同じく、スペースで停止、エスケープで抜けます
  8:                         JUN.16th.1993 (ats)*/
  9: #include"basic.h"
 10: #include"iocslib.h"
 11:
 12: /* 三角形の変形先の座標バッファ */
 13: struct  TRIANGLE {
 14:         int x1,y1,x2,y2,x3,y3; }
 15:         oncub[12][60],
 16:         ongra[12] = {
 17:         256,256,384,384,256,384, 384,256,384,384,256,256 };
 18: /* 直方体の座標 */
 19: int     cube[8][3] = {
 20:         -50, 50,-50, 50, 50,-50,
 21:         -50, 50, 50, 50, 50, 50,
 22:         -50,-50,-50, 50,-50,-50,
 23:         -50,-50, 50, 50,-50, 50 };
 24: /* 座標の連結情報 */
 25: int     cbcon[12][3] = {
 26:         2,1,0,3,1,2, 4,0,1,5,4,1,
 27:         3,5,1,7,5,3, 4,2,0,6,2,4,
 28:         7,3,2,6,7,2, 7,4,5,6,4,7 };
 29: /* 各三角形の表示・非表示のフラグ */
 30: int     flag[12][60];
 31: void    draw();
 32: void    bkdraw();
 33: void    rot();
 34: extern  trtrfm();
 35: extern  dwait();
 36: main()
 37: {
 38:  int i,j,k,l,r1 = 0,r2 = 0;
 39:  int po[8][3],rp[8][2];
 40:  int ax,ay,az,bx,by,bz,cx,cy,cz;
 41:  char key;
 42:    /* 前準備 */
 43:    cls();
 44:    console( 0,31,0 );
 45:    screen( 0,3,1,1 );
 46:    window( 0,0,511,511 );
 47:    OS_CUROF();
 48:    for( i = 0; i < 60; i++ )
 49:    {
 50:      /* 座標を回転 */
 51:      for( j = 0; j != 8; j++ )
 52:      {
 53:        rot( &po[j],&cube[j],i );
 54:        /* 平面投射 */
 55:        rp[j][0] = 128+po[j][2]*500/(po[j][0]+900);
 56:        if( ( i % 2 ) == 0 )
 57:          rp[j][1] = 128-po[j][1]*500/(po[j][0]+900);
 58:        else
 59:          rp[j][1] = 384-po[j][1]*500/(po[j][0]+900);
 60:      }
 61:      /* 三角形割り当て(先月とあまり変わらない) */
 62:      for( j = 0; j < 12; j++ )
 63:      {
 64:        oncub[j][i].x1 = rp[cbcon[j][0]][0];
 65:        oncub[j][i].y1 = rp[cbcon[j][0]][1];
 66:        oncub[j][i].x2 = rp[cbcon[j][1]][0];
 67:        oncub[j][i].y2 = rp[cbcon[j][1]][1];
 68:        oncub[j][i].x3 = rp[cbcon[j][2]][0];
 69:        oncub[j][i].y3 = rp[cbcon[j][2]][1];
 70:        ax = po[cbcon[j][0]][0]-po[cbcon[j][1]][0];
 71:        ay = po[cbcon[j][0]][1]-po[cbcon[j][1]][1];
 72:        az = po[cbcon[j][0]][2]-po[cbcon[j][1]][2];
 73:        bx = po[cbcon[j][2]][0]-po[cbcon[j][1]][0];
 74:        by = po[cbcon[j][2]][1]-po[cbcon[j][1]][1];
 75:        bz = po[cbcon[j][2]][2]-po[cbcon[j][1]][2];
 76:        cx = ay*bz-az*by;
 77:        cy = az*bx-ax*bz;
 78:        cz = ax*by-ay*bx;
 79:        ax = po[cbcon[j][1]][0] + 1000;
 80:        ay = po[cbcon[j][1]][1];
 81:        az = po[cbcon[j][1]][2];
 82:        /* つぎに内積を取って面の方向を見る */
 83:        if( ax*cx+ay*cy+az*cz < 0 )
 84:          flag[j][i] = 1;
 85:        else
 86:          flag[j][i] = 0;
 87:      }
 88:    }
 89:    bkdraw();
 90:    i = 0;
 91:    draw( 0 );
 92:    do
 93:    {
 94:      dwait();
 95:      if( ( i % 2 ) == 0 )
 96:         home( 0,0,0 );
 97:       else
 98:         home( 0,0,256 );
 99:      i++;
100:      if( i > 59 )
101:        i = 0;
102:      draw( i );
103:      key = BITSNS( 6 );
104:      if( ( key & 0x20 ) != 0 )
105:        while( ( key & 0x20 ) != 0 )
106:          key = BITSNS( 6 );
107:      key = BITSNS( 0 );
108:    }
109:    while( (key & 2) == 0 );
110:    console( 0,31,1 );
111:    OS_CURON();
112:    return( 0 );
113:  }
114:
115: void draw( i )
116: /* 描く */
117:  int i;
118:  {
119:  int j;
120:    if( ( i % 2 ) == 0 )
121:      fill( 70, 70,185,185,0 );
122:    else
123:      fill( 70,326,185,441,0 );
124:    for( j = 0; j < 12; j++ )
125:      if( flag[j][i] == 0 )
126:        trtrfm( &ongra[(j % 2)],&oncub[j][i] );
127:  }
128:
129: void bkdraw()
130: /* 背景を書く */
131:  {
132:    int i,j;
133:    fill( 256,256,511,511,0xffff );
134:    for( j = 0; j < 4; j++ )
135:      for( i = 0; i < 4; i++ )
136:      {
137:        circle( i*64+288,j*64+288,20,rgb( 31,0,0 ),0,360,256 );
138:        paint( i*64+288,j*64+288,rgb( 31,0,0 ) );
139:      }
140:  }
141:
142: void rot( p,ip,par )
143: /* 入力座標ipよりpに
144:    parにしたがった角度分回転した座標を返す */
```

```
145:  int (*p)[3],(*ip)[3],par;
146:  {
147:  double tmp,si,co;
148:          si = sin( (double)par/15*3.1415 );
149:          co = cos( (double)par/15*3.1415 );
150:          tmp  = (*ip)[0]*si+(*ip)[1]*co;
151:          (*p)[0] = (*ip)[0]*co-(*ip)[1]*si;
152:          si = sin( (double)par/30*3.1415 );
153:          co = cos( (double)par/30*3.1415 );
154:          (*p)[1] = (*ip)[2]*si+tmp*co;
155:          (*p)[2] = (*ip)[2]*co-tmp*si;
156:  }
157:  /* (*):徹夜明けでハイになっている */
```

リスト2

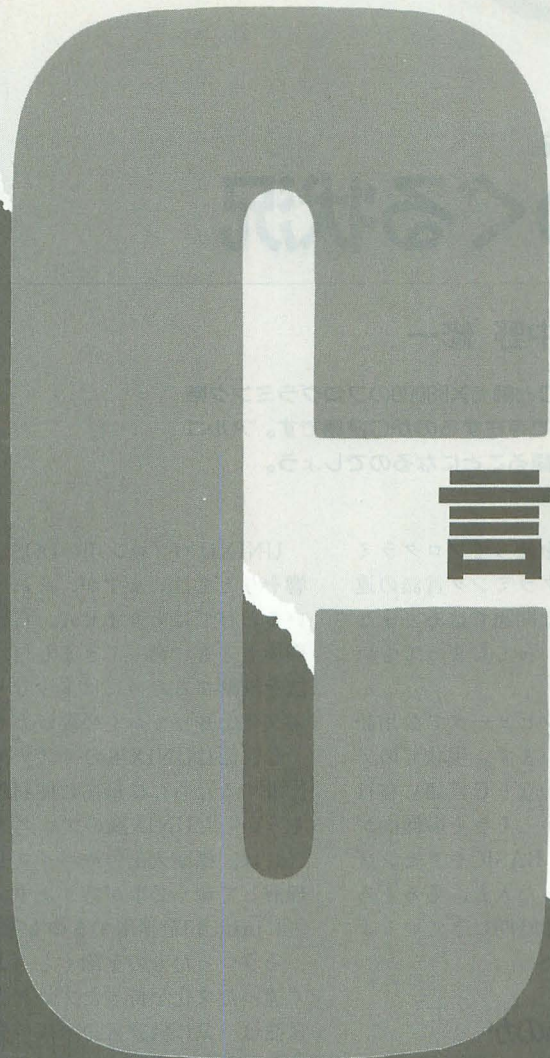
```
  1: **********************************************
  2: *
  3: *   trtrfm( pre_p,aft_p  )
  4: *         三角形どうしの自由変形
  5: *
  6: *  ≡引数≡      ≡ 引数の意味 ≡
  7: *  pre_p : 変形前の三角形頂点のポインタ
  8: *  aft_p : 変形後の三角形頂点のポインタ
  9: *     なお頂点の構造体は、
 10: *       struct POINTS = {
 11: *         int x1,y1,x2,y2,x3,y3 };
 12: *       とでも定義しておいたらええわいな
 13: *                って先月も書いたよな……
 14: *
 15: **********************************************
 16:
 17: *_B_SUPER        equ       $81
 18:
 19: include iocscall.mac
 20:
 21:          .xdef _trtrfm
 22:          .xdef _dwait
 23:
 24:          .text
 25:          .even
 26:
 27:          * IOCSコールのマクロ
 28: *IOCS    macro   callname
 29: *        moveq.l #callname,d0
 30: *        trap    #15
 31: *        endm
 32: set_pr1  macro r0,r1,r2,r3
 33:          local   minus,out_of_sp
 34:          * パラメータ設定のマクロ
 35:          * a4,a5にそれぞれ始点、終点の先頭番地を入れとく
 36:          * かならず終点の方が下に位置していること
 37:          move.l  4(a5),r0
 38:          sub.l   4(a4),r0            * 2点のY軸の差
 39:          move.l  (a5),r1
 40:          sub.l   (a4),r1             * 2点のX軸の差
 41:          tst.l   r0
 42:          ble     out_of_sp
 43:          tst.l   r1
 44:          bmi     minus
 45:          * 増加率を計算する
 46:          move.l  r1,r2
 47:          divs    r0,r2
 48:          move.w  r2,r3
 49:          ext.l   r2                  * 有効幅を32ビットに戻す
 50:          mulu    r0,r3
 51:          sub.l   r3,r1
 52:          bra     out_of_sp
 53: minus:
 54:          move.l  #0,r2
 55:          sub.l   r1,r2
 56:          move.l  r2,r1
 57:          divs    r0,r2
 58:          move.w  r2,r3
 59:          ext.l   r2                  * 有効幅を32ビットに戻す
 60:          mulu    r0,r3
 61:          sub.l   r3,r1
 62:          move.l  #0,r3
 63:          sub.l   r1,r3
 64:          move.l  r3,r1
 65:          move.l  #0,r3
 66:          sub.l   r2,r3
 67:          move.l  r3,r2
 68: out_of_sp:
 69:          endm
 70:          * マクロから抜けた時点で
 71:          * r0にはY方向の増分、r1にはX方向の整数以下部分の増分
 72:          * r2にはX方向の整数部分の増分
 73:          * が入っている
 74: flw_edge macro   r0,r1,r2,r3,r4
 75:          local   minus,non_incl
 76:          * 変形先の三角形の輪郭にそって座標を動かしていくマクロ
 77:          * r0はXの整数以下の増分、r1は整数部の増分
 78:          * r2はY方向の増分、r3はX座標
 79:          * r4は増分用のカウンタ
 80:          * の入っているレジスタ、あるいはメモリアドレス
 81:          add.l   r1,r3
 82:          cmpa.l  #0,r0
 83:          blt     minus
 84:          add.l   r0,r4
 85:          cmp.l   r2,r4
 86:          blt     non_incl
 87:          * 整数以下が範囲を越えたのでX座標を増やす
 88:          sub.l   r2,r4
 89:          addq.l  #1,r3
 90:          bra     non_incl
 91: minus:
 92:          sub.l   r0,r4
 93:          cmp.l   r2,r4
 94:          blt     non_incl
 95:          * 整数以下が範囲を越えたのでX座標を減らす
 96:          sub.l   r2,r4
 97:          subq.l  #1,r3
 98: non_incl:
 99:          endm
100: set_pr2 macro
101:          * (a4),(a5)で指定された始点、終点の間を、
102:          * d0回の増減で動かせるようにパラメーターを
103:          * 設定するマクロ
104:          * まずX座標に関する操作をする
105:          move.l  (a4),d2
106:          move.l  (a5),d1             * 始点と終点のX座標
107:          sub.l   d2,d1
108:          move.l  d1,d2
109:          divs    d0,d1               * X方向の整数範囲の増分
110:          ext.l   d1                  * 有効範囲を倍加
111:          move.l  d1,d3
112:          muls    d0,d3
113:          sub.l   d3,d2               * X方向少数部分の増分
114:          * つぎにY座標に関する操作をする
115:          move.l  4(a4),d4
116:          move.l  4(a5),d3            * 始点と終点のY座標
117:          sub.l   d4,d3
118:          move.l  d3,d4
119:          divs    d0,d3               * Y方向の整数範囲の増分
120:          ext.l   d3                  * 有効範囲を倍加
121:          move.l  d3,d5
122:          muls    d0,d5
123:          sub.l   d5,d4               * Y方向少数部分の増分
124:          endm
125:          * マクロから抜けた時点で
126:          * d1,d2にはXの整数増分、少数増分が
127:          * d3,d4にはYの整数増分、少数増分が入っている
128: set_pr3 macro
129:          * set_pr2 とほとんど同じ
130:          * 座標がロングワードからワードになっただけ
131:          move.w  (a4),d2
132:          move.w  (a5),d1             * 始点と終点のX座標
133:          ext.l   d2
134:          ext.l   d1
135:          sub.l   d2,d1
136:          move.l  d1,d2
137:          divs    d0,d1               * X方向の整数範囲の増分
138:          ext.l   d1                  * 有効範囲を倍加
139:          move.l  d1,d3
140:          muls    d0,d3
141:          sub.l   d3,d2               * X方向少数部分の増分
142:          * つぎにY座標に関する操作をする
143:          move.w  2(a4),d4
144:          move.w  2(a5),d3            * 始点と終点のY座標
145:          ext.l   d3
146:          ext.l   d4
147:          sub.l   d4,d3
148:          move.l  d3,d4
149:          divs    d0,d3               * Y方向の整数範囲の増分
150:          ext.l   d3                  * 有効範囲を倍加
151:          move.l  d3,d5
152:          muls    d0,d5
153:          sub.l   d5,d4               * Y方向少数部分の増分
154:          endm
155: move_point macro
156:          local   non_frc_x,minus_x,non_frc_y,minus_y
157:          * 上のマクロで計算済みのパラメーターを駆使して
158:          * 2点間を指定段階で等分しつつ点を動かすマクロ
159:          * 上のマクロの結果の他
160:          * d5,d6をX,Y方向のカウンタ
161:          * d0,d7を座標値、a0はカウンタのしきい値として利用する
162:          * X座標に関する処理
163:          add.l   d1,d0               * 整数増分を座標に足す
164:          tst.l   d2
165:          blt     minus_x
166:          add.l   d2,d5               * カウンタに少数増分を足す
167:          cmp.l   a0,d5               * しきい値とカウンタを比較
168:          blt     non_frc_x
169:          sub.l   a0,d5
170:          addq.l  #1,d0
171:          bra     non_frc_x
172: minus_x:
173:          sub.l   d2,d5               * カウンタに少数増分を足す
174:          cmp.l   a0,d5               * しきい値とカウンタを比較
175:          blt     non_frc_x
176:          sub.l   a0,d5
177:          subq.l  #1,d0
178: non_frc_x:
179:          * Y座標に関する処理
180:          add.l   d3,d7               * 整数増分を座標に足す
181:          tst.l   d4
182:          blt     minus_y
183:          add.l   d4,d6               * カウンタに少数増分を足す
184:          cmp.l   a0,d6               * しきい値とカウンタを比較
```

▶やっとうちのX1turboIIにFM音源ボードがついた。それも新品で……。しかし,値段は昔のままで安くなっていなかった。　齋藤 眞二(19)東京都

```
185:         blt      non_frc_y
186:         sub.l    a0,d6
187:         addq.l   #1,d7
188:         bra      non_frc_y
189: minus_y:
190:         sub.l    d4,d6               * カウンタに少数増分を足す
191:         cmp.l    a0,d6               * しきい値とカウンタを比較
192:         blt      non_frc_y
193:         sub.l    a0,d6
194:         subq.l   #1,d7
195: non_frc_y:
196:         endm
197:
198: _trtrfm
199:         link     a6,#-128
200:         movem.l  d1-d7/a0-a6,-(sp)
201:         clr.l    a1
202:         IOCS     _B_SUPER            * スーパーバイザモードに移行
203:         move.l   d0,-(sp)
204:         lea.l    after(pc),a5
205:         movea.l  12(a6),a4
206:         move.l   (a4)+,(a5)+
207:         move.l   (a4)+,(a5)+
208:         move.l   (a4)+,(a5)+
209:         move.l   (a4)+,(a5)+
210:         move.l   (a4)+,(a5)+
211:         move.l   (a4)+,(a5)+
212:         lea.l    previous(pc),a5
213:         movea.l  8(a6),a4
214:         move.l   (a4)+,(a5)+
215:         move.l   (a4)+,(a5)+
216:         move.l   (a4)+,(a5)+
217:         move.l   (a4)+,(a5)+
218:         move.l   (a4)+,(a5)+
219:         move.l   (a4)+,(a5)+
220:         * 手始めに、3つの頂点を上から順に並ぶようソーティングする
221: top_of_sorting:
222:         lea.l    after(pc),a0
223:         lea.l    previous(pc),a1
224:         move.l   4(a0),d0            * 1番目の点のY座標
225:         move.l   12(a0),d1           * 2番目の点のY座標
226:         cmp.l    d1,d0       * 1番目と2番目のY座標の値を比べる
227:         ble      non_swap_1_2
228:         * 1番と2番を入れ替える
229:         move.l   8(a0),d2
230:         move.l   12(a0),d3
231:         move.l   0(a0),8(a0)
232:         move.l   4(a0),12(a0)
233:         move.l   d2,0(a0)
234:         move.l   d3,4(a0)
235:         exg.l    d1,d0
236:         move.l   8(a1),d2
237:         move.l   12(a1),d3
238:         move.l   0(a1),8(a1)
239:         move.l   4(a1),12(a1)
240:         move.l   d2,0(a1)
241:         move.l   d3,4(a1)
242: non_swap_1_2:
243:         * この時点で、1番は2番より上にある
244:         cmp.l    20(a0),d1           * 2番目と3番目を比べる
245:         ble      out_of_sorting      * めでたくソーティング終了
246:         * 2番と3番を入れ替える
247:         move.l   16(a0),d2
248:         move.l   20(a0),d3
249:         move.l   8(a0),16(a0)
250:         move.l   12(a0),20(a0)
251:         move.l   d2,8(a0)
252:         move.l   d3,12(a0)
253:         move.l   16(a1),d2
254:         move.l   20(a1),d3
255:         move.l   8(a1),16(a1)
256:         move.l   12(a1),20(a1)
257:         move.l   d2,8(a1)
258:         move.l   d3,12(a1)
259:         * 入れ替えた2番と1番も調べる
260:         cmp.l    12(a0),d0
261:         ble      out_of_sorting
262:         * 1番と2番を入れ替える
263:         move.l   8(a0),d2
264:         move.l   12(a0),d3
265:         move.l   0(a0),8(a0)
266:         move.l   4(a0),12(a0)
267:         move.l   d2,0(a0)
268:         move.l   d3,4(a0)
269:         move.l   8(a1),d2
270:         move.l   12(a1),d3
271:         move.l   0(a1),8(a1)
272:         move.l   4(a1),12(a1)
273:         move.l   d2,0(a1)
274:         move.l   d3,4(a1)
275: out_of_sorting:
276:         lea.l    lr_flag(pc),a2
277:         move.w   #0,(a2)
278:         * まず変形先の座標をバッファに書き出す
279:         move.l   4(a0),d0
280:         cmp.l    12(a0),d0
281:         bne      leap_ds_ex
282:         cmp.l    20(a0),d0
283:         beq      exception           * 3点とも同じ高さにあった
284: leap_ds_ex:
285:         * 1番から2番に動かすためのパラメーター類
286:         movea.l  a0,a4
287:         movea.l  a0,a5
288:         addq.l   #8,a5
289:         set_prl  d4,d5,d7,d2         * パラメーター計算のマクロ
290:         * 1番から3番に動かすためのパラメーター類
291:         addq.l   #8,a5
292:         set_prl  d0,d1,d3,d6         * パラメーターを計算
293:         move.l   (a0),d2             * X軸の初期値を取り出す
294:         move.l   d2,d6
295:         move.l   d0,a1
296:         move.l   d4,a6
297:         movea.l  d1,a2
298:         movea.l  d5,a3
299:         move.l   d0,d1
300:         asr.l    d1
301:         move.l   d4,d5
302:         asr.l    d5
303:         tst.l    d4
304:         bne      leap_following11
305:         move.l   8(a0),d6
306:         lea.l    a_buff(pc),a0
307:         bne      leap_following12
308: leap_following11:
309:         lea.l    a_buff(pc),a0
310: top_of_following1:
311:         move.w   d6,(a0)+
312:         move.w   d2,(a0)+            * 座標をバッファに書き込む
313:         * 汎用のマクロを呼び出す
314:         flw_edge a2,d3,a1,d2,d1
315:         flw_edge a3,d7,a6,d6,d5
316:         subq.l   #1,d0
317:         subq.l   #1,d4
318:         bne      top_of_following1
319: leap_following12:
320:         cmp.l    d2,d6
321:         blt      rightway
322:         lea.l    lr_flag(pc),a6
323:         move.w   #1,(a6)
324: rightway:
325:         tst.l    d0
326:         beq      out_of_following1
327:         addq.l   #8,a4
328:         set_prl  d4,d5,d7,d6
329:         lea.l    after(pc),a6
330:         move.l   8(a6),d6
331:         move.l   d4,a6
332:         movea.l  d5,a3
333:         move.l   d4,d5
334:         asr.l    d5
335:         addq.l   #1,d0
336: top_of_following12:
337:         move.w   d6,(a0)+
338:         move.w   d2,(a0)+            * 座標をバッファに書き込む
339:         flw_edge a2,d3,a1,d2,d1
340:         flw_edge a3,d7,a6,d6,d5
341:         subq.l   #1,d0
342:         bne      top_of_following12
343:         bra      out_of_following1
344: exception:
345:         * 3つの点がすべて同じ高さにあった時の処理
346:         * まずいちばん左の点を見つける
347:         move.l   (a0),d0
348:         cmp.l    8(a0),d0
349:         blt      through_ex1
350:         move.l   8(a0),d0
351: through_ex1:
352:         cmp.l    20(a0),d0
353:         blt      through_ex2
354:         move.l   20(a0),d0
355: through_ex2:
356:         * つぎにいちばん右の点を見つける
357:         move.l   (a0),d1
358:         cmp.l    8(a0),d1
359:         bgt      through_ex3
360:         move.l   8(a0),d1
361: through_ex3:
362:         cmp.l    20(a0),d1
363:         bgt      through_ex4
364:         move.l   20(a0),d1
365: through_ex4:
366:         lea.l    a_buff(pc),a1
367:         move.l   d0,(a1)+
368:         move.l   d1,(a1)+
369: out_of_following1:
370:         * つぎに変形元の座標をバッファに書き出す
371:         * 1番目の頂点から2番目に向かう線の処理
372:         lea.l    previous(pc),a4
373:         move.l   a4,a5
374:         adda.l   #8,a5
375:         lea.l    after(pc),a3
376:         move.l   12(a3),d0
377:         sub.l    4(a3),d0
378:         addq.l   #1,d0
379:         set_pr2
380:         movea.l  d0,a0               * しきい値をa0に代入
381:         movea.l  a0,a2
382:         move.l   d0,d5
383:         asr.l    d5
384:         move.l   d5,d6
385:         move.l   (a4),d0
386:         move.l   4(a4),d7            * 座標値を代入
387:         lea.l    p_buf1(pc),a1
388: top_of_following21:
389:         move_point                   * マクロを呼び出す
390:         move.w   d0,(a1)+
391:         move.w   d7,(a1)+            * 座標値をバッファにため込む
392:         subq.l   #1,a2
393:         cmpa.l   #0,a2
394:         bne      top_of_following21
```

▶X68030を買おうか，X68000 Compact RED ZONEを買おうか悩んでいます。スピードをとるか互換性をとるか……金銭的にはどちらでもいいんだけど。鈴木　健也(26)静岡県

```
395:            * 2番目の頂点から3番目に向かう線の処理
396:            adda.l  #8,a4
397:            adda.l  #8,a5
398:            move.l  20(a3),d0
399:            sub.l   12(a3),d0
400:            beq     out_of_following22
401:            addq.l  #1,d0
402:            set_pr2
403:            movea.l d0,a0                   * しきい値をa0に代入
404:            movea.l a0,a2
405:            move.l  d0,d5
406:            asr.l   d5
407:            move.l  d5,d6
408:            move.l  (a4),d0
409:            move.l  4(a4),d7                * 座標値を代入
410:            cmpa.l  #0,a2
411:            beq     out_of_following22
412: top_of_following22:
413:            move_point                      * マクロを呼び出す
414:            move.w  d0,(a1)+
415:            move.w  d7,(a1)+                * 座標値をバッファにため込む
416:            subq.l  #1,a2
417:            cmpa.l  #0,a2
418:            bne     top_of_following22
419:            * 1番目の頂点から3番目に向かう線の処理
420: out_of_following22:
421:            lea.l   previous(pc),a4
422:            move.l  a4,a5
423:            adda.l  #16,a5
424:            move.l  20(a3),d0
425:            sub.l   4(a3),d0
426:            addq.l  #1,d0
427:            set_pr2
428:            movea.l d0,a0                   * しきい値をa0に代入
429:            movea.l a0,a2
430:            move.l  d0,d5
431:            asr.l   d5
432: move.l #0,d5
433:            move.l  d5,d6
434:            move.l  (a4),d0
435:            move.l  4(a4),d7                * 座標値を代入
436:            lea.l   p_buf2(pc),a1
437: top_of_following23:
438:            move_point                      * マクロを呼び出す
439:            move.w  d0,(a1)+
440:            move.w  d7,(a1)+                * 座標値をバッファにため込む
441:            subq.l  #1,a2
442:            cmpa.l  #0,a2
443:            bne     top_of_following23
444:            * バッファ内の頂点データをもとに
445:            * 三角形を画面に描き出す
446:            lea.l   p_buf1(pc),a4
447:            lea.l   p_buf2(pc),a5
448:            lea.l   a_buff(pc),a1
449:            lea.l   lr_flag(pc),a2
450:            cmpi.w  #1,(a2)
451:            beq     leftway
452:            * 右向きに描画する場合
453:            lea.l   after(pc),a2
454:            move.l  4(a2),d0
455:            move.l  20(a2),d1
456:            sub.l   d0,d1
457: *          addq.l  #1,d1
458:            move.l  d1,a3                   * 描画する三角形のY増分
459:            asl.l   #8,d0                   * 描画開始ラスターアドレスを求める
460:            asl.l   #2,d0
461:            addi.l  #$c00000,d0
462: top_of_tridraw_r:
463:            move.w  (a1),d1
464:            ext.l   d1
465:            asl.l   #1,d1
466:            add.l   d1,d0                   * 描画開始のアドレスを求める
467:            movea.l d0,a6
468:            move.w  (a1)+,d1
469:            move.w  (a1)+,d0
470:            sub.w   d1,d0
471:            ext.l   d0
472:            addq.l  #1,d0
473:            beq     out_of_trtrfm
474:            set_pr3
475:            move.l  d0,a0                   * しきい値を代入
476:            move.l  d0,a2
477:            move.l  d0,d5
478:            asr.l   d5
479:            move.l  d5,d6
480:            move.w  (a4)+,d0
481:            move.w  (a4)+,d7                * 座標を代入
482:            ext.l   d0
483:            ext.l   d7
484:            adda.l  #4,a5
485: top_of_raster_r:
486:            * 変形先の三角形の横一列を書き出す
487:            move_point                      * 点を動かすマクロ
488:            movem.l d0-d1/a0,-(sp)
489:            move.l  d7,d1
490:            asl.l   #8,d1
491:            asl.l   #2,d1                   * ラスターアドレスを求める
492:            asl.l   #1,d0
493:            add.l   d0,d1
494:            addi.l  #$c00000,d1
495:            movea.l d1,a0
496:            move.w  (a0),(a6)+
497:            movem.l (sp)+,d0-d1/a0
498:            subq.l  #1,a2
499:            cmpa.l  #0,a2
500:            bne     top_of_raster_r
501:            move.l  a6,d0
502:            andi.l  #$fffc00,d0
503:            addi.l  #1024,d0
504:            subq.l  #1,a3
505:            cmpa.l  #0,a3
506:            bgt     top_of_tridraw_r
507:            bra     out_of_trtrfm
508: leftway:
509:            * 左向きに描画する場合
510:            lea.l   after(pc),a2
511:            move.l  4(a2),d0
512:            move.l  20(a2),d1
513:            sub.l   d0,d1
514:            addq.l  #1,d1
515:            move.l  d1,a3                   * 描画する三角形のY増分
516:            asl.l   #8,d0                   * 描画開始ラスターアドレスを求める
517:            asl.l   #2,d0
518:            addi.l  #$c00000,d0
519: top_of_tridraw_l:
520:            move.w  (a1),d1
521:            ext.l   d1
522:            asl.l   #1,d1
523:            addq.l  #2,d1
524:            add.l   d1,d0                   * 描画開始のアドレスを求める
525:            movea.l d0,a6
526:            move.w  (a1)+,d0
527:            move.w  (a1)+,d1
528:            sub.w   d1,d0
529:            ext.l   d0
530:            addq.l  #1,d0
531:            beq     out_of_trtrfm
532:            set_pr3
533:            move.l  d0,a0                   * しきい値を代入
534:            move.l  d0,a2
535:            move.l  d0,d5
536:            asr.l   d5
537:            move.w  (a4)+,d0
538:            move.w  (a4)+,d7                * 座標を代入
539:            ext.l   d0
540:            ext.l   d7
541:            adda.l  #4,a5
542: top_of_raster_l:
543:            * 変形先の三角形の横一列を書き出す
544:            move_point                      * 点を動かすマクロ
545:            movem.l d0-d1/a0,-(sp)
546:            move.l  d7,d1
547:            asl.l   #8,d1
548:            asl.l   #2,d1                   * ラスターアドレスを求める
549:            asl.l   #1,d0
550:            add.l   d0,d1
551:            addi.l  #$c00000,d1
552:            movea.l d1,a0
553:            move.w  (a0),-(a6)
554:            movem.l (sp)+,d0-d1/a0
555:            subq.l  #1,a2
556:            cmpa.l  #0,a2
557:            bgt     top_of_raster_l
558:            move.l  a6,d0
559:            andi.l  #$fffc00,d0
560:            addi.l  #1024,d0
561:            subq.l  #1,a3
562:            cmpa.l  #0,a3
563:            bne     top_of_tridraw_l
564: out_of_trtrfm:
565:            move.l  (sp)+,a1
566:            IOCS    _B_SUPER                * スーパーバイザモードから抜ける
567:            move.l  a2,d0
568:            movem.l (sp)+,d1-d7/a0-a6
569:            unlk    a6
570:            rts
571:
572: *********************************************
573: *      dwait( plane )
574: *  垂直帰線時間まで待つ。先月と同一物
575: *********************************************
576:
577: _dwait
578:            move.l  a1,-(sp)
579:            clr.l   a1
580:            IOCS    _B_SUPER                * スーパーバイザモードへ
581: loop_dw:
582:            btst    #4,$e88001
583:            bne     loop_dw
584:            move.l  d0,a1
585:            IOCS    _B_SUPER
586:            move.l  (sp)+,a1
587:            rts
588:
589:            * ここからデータ領域
590:            .even
591:            .data
592: lr_flag:
593:            .ds.l   1
594: after:
595:            .ds.l   6
596: previous:
597:            .ds.l   6
598: a_buff
599:            .ds.w   512
600: p_buf1
601:            .ds.w   512
602: p_buf2
603:            .ds.w   512
```

[特集]

C言語実践的入門

UNIX生まれのC言語はいまやパソコンの標準的なプログラミング言語となった。これは同時に，ミニコンやワークステーションはもちろん，パソコンから大型コンピュータ，スーパーコンピュータ，はてはDSPまで，ありとあらゆるコンピュータの共通言語となったともいえる。特にRISCマシンではC言語なしにプログラム開発は考えられないといってもいいだろう。

「高級アセンブラ」とか「低級言語」と呼ばれていた頃に比べ，現在のC言語体系は肥大化，複雑化してきている。当然，その位置づけも，プロフェッショナルのためのシステム記述言語であった時代と比べるとかなり変わり，ライブラリは必ずエラーチェックを行うといった低級言語らしからぬ仕様がまかり通るほどに大衆化してしまっている。

そして，あいかわらず自由度は高いものの，「暴走なんかは当たり前」といったプログラミングもだんだん許容されなくなってきた。

巨大な高級言語としてのC言語，あいかわらずの低級言語としてのC言語。それはもちろん，それを使ってプログラムを作成する者の姿勢に依存するものであろう。どのような使い方であれ，この「公用語」たるC言語を無視してはプログラムを語れない時代がやってきつつある。

CONTENTS

概論

C言語をめぐる状況

Nakano Shuichi 中野 修一

アセンブラを頂点とし，C言語，BASICと続くX68000のプログラミング階層構造。このなかでももっとも広い範囲で存在するのがC言語です。アルゴリズムを語るとき，これからはC言語で語ることになるのでしょう。

公用語としてのC

最近はプログラミング言語の話題が少なくなってきました。「製品」に対する話題はまだあるのですが，「言語」自体への関心が低くなったように思われます。言語開発の華やかなりし頃に比べるとなんとも寂しいかぎりです。隆盛をきわめたCOBOLやFORTRANもいまは前世代の遺産を守るために使われるのみ。新しいパラダイムに対する開発はもっぱらC言語（かC++）で行われるようになっています。

「クリーンコンピュータ（死語）」という言葉からパソコンに入った私などは，まんまと「クリーンコンピュータだからいろんな言語が使えるんだ。へぇ」と感心していていたものでした。以来，正直にいろんな言語を見てきましたが，個人的にはC言語はあまり好きな言語ではありません。

Cは生まれたときにははっきりした方針を持っていた言語なのですが，その後，時代の要請とともにわりと普通の言語っぽくなったような気がします。もちろん言語仕様はほとんど変わっていないのですが，処理系の作り方やプログラミングの方法論がより高級言語化してきたことによるものです。普通の言語っぽいということは，それはそれで多分よいことなのでしょう。

世間一般で考えると，X68000ではユーザー数に対するC compiler PRO-68Kの売れ具合は異常といっていいでしょう。

しかし，その割にはC言語によるプログラム開発を行う人がそんなには多くないのも事実です。

もちろん，そのすべてがC言語を使っているかというとそうでもないはずですし，BASICコンパイラとしての用途はありますし，マニュアルやアセンブラなどを入手するために買った人だっていると思いますが。

実際問題として，BASICなりアセンブラなり，なんらかの言語を使ってプログラミングができれば，プログラミング言語の違いなどはそれほど大きな問題ではありません。単に慣れの問題といってしまってもいいかもしれません。

C言語はあらゆるコンピュータで公用語としての性格を強めています。現状でのアルゴリズム解説などはかなりC言語に移行していますし，今後はいっそうその傾向が強まっていくでしょう。BASICやアセンブラで不自由していなかった人も，そろそろC言語をたしなんでおく時期にきているように思われます。

なぜ敷居が高いのか

C言語を学ぼうとしてつまずく人の多さはほかの言語の比ではないでしょう。障害として有名なところで，たとえば，構造体，ポインタというものですが，それらの概念自体はそう特殊なものではないように思われます。

構造体は異なった型のデータの混ざった一次元配列ですし，データベースの基本構造などを知っている人なら簡単に察しがつくでしょう。ポインタは配列の添字と配列名を一緒にした変数の形態と思えばそう間違ってはいないでしょう。

ある程度なんらかのプログラミング経験がある人にとっては，それほど大きな障害ではないと思われます。にもかかわらず，C言語というのはどうも敷居が高い言語なのです。

●メモリのイメージを必要とする

概念ではそれほど難しくない構造体やポインタでも，使いこなすには頭の中にある程度，使用中のメモリのイメージを持っておく必要があります。それとCPUのアドレッシングモードはひととおり理解しておいたほうがいいでしょう。

●UNIX環境を理解する必要がある

UNIXはパソコン用のDOSにも大きな影響を与えてはいますが，それほど徹底しているわけではありません。C言語はUNIX環境とともに育ってきました。C言語の作法を理解するためには多少なりともUNIXの文化を知っておく必要があります。

C言語はUNIX風のアプリケーションを作成するためには最高に便利な言語です。そういったUNIX風のプログラムを読むときには，標準入出力やパイプという概念を理解しておく必要があります。なにも考えずにprintf()を多用するのも考えものです。

そういったものを除くと（またはまったく違った文化を持ったプログラムでは），C言語は一気に低級言語の様相を呈してくることもあります（ライブラリの問題）。

●ハナモゲラであること

UNIXではタイプ量を減らすためにコマンドなどはかなり簡略されたハナモゲラなものになっています。それを受けてか，C言語で使われる関数の名前もハナモゲラになっています。ある種の規則さえわかればそう難しくはないのですが，初心者には一見しただけではなにをするものなのかわからないプログラムというのもあるでしょう。

●とりあえず触ることが難しい

C言語はBASICなどと違ってコンパイラ言語です。ですからC言語のプログラムは誌面に掲載されていても，プログラムのすべてを打ち込んでコンパイルしなければ結果が出てきません。それだけならまだしも，BASICに比べるとプログラム自体も長くなる傾向にあります。

とりあえず触ってみること，という意味ではGCCのような高性能なコンパイラよりもXC ver.1のような軽いコンパイラのほうがありがたい場面もあるでしょう。

●前置きが多い

C言語の支持者に，C言語は言語仕様が小さいから覚えることが少なくていい，といった人がいます。正論です。こういう人

には，RISCプロセッサはさぞかしマシン語が使いやすいCPUなのでしょう。

C言語の言語仕様が小さいにもかかわらず，C compiler PRO-68Kのパッケージがあんなに重いのは，つまるところ「言語仕様外の必要事項が山のようにある」ということを意味しています。ひねくれたいい方をすれば，C言語で画面に“Hello World.”と表示するにはハードウェアの知識が不可欠になります（もしくは不可能）。

とりあえず，“#include <stdio.h>”のような定型文は必須のものです。C言語でいきなりプログラムを書き始めることができないのは，こういった前置きがないとなにもできないからです。

Cプログラマであればエディタを立ち上げた途端，自動的に指がこれらの呪文を弾き出していきます。おそらく誰も不自由だとは感じていないのでしょう。

●ライブラリが多い

ということで，ライブラリがないとなにもできない（普通の人には）わけですが，なんやかんやでライブラリが膨大な数になります。多いということは悪いことではないのですが，なかにはほとんど同じ意味のライブラリなどもあり，どれを選んで使うべきかということがわかりにくくなっています。ちなみに，XCのマニュアルでは基本的にアルファベット順に並んでいます。

結局，どの状況でどのライブラリを使ってやるのがいいのかは，他人のプログラムを参考にするのがいちばんなのでしょう。

＊　＊　＊

以上のような理由からC言語は敷居の高い言語として君臨しているわけです。さらに加えるなら自由度が高すぎることが，わかりにくさを増しているようにも思えます。書式や字下げのクセ，関数の大きさなどにも個人差が大きく出てきます。特に上級者はテクニカルな書き方が多いようです。他人のプログラムが読みにくいというのはやはり障害になります。

自由度はシステムの最大の長所でもあります。しかし，ポインタのわかりにくさの半分くらいは整数とポインタが等価だというところにありそうな気もするのですが。

GCCについて

X68000ユーザーには多くのアセンブラユーザーがいます。せっかくの68000マシンならばアセンブラを使うのがいちばんなのかもしれません。プログラミングの最難関として存在する「マシン語」ですが，極端な話，C言語よりはるかに簡単だという人も少なくありません。

ポインタなどというもってまわった用語で惑わされることもありませんし，分厚いマニュアルも必要ありません（ハード関係のマニュアルは必要ですが）。

それでもかなりの上級者までがC言語を使うのはやはりGCCの力によるところが大きいといえるでしょう。

ライブラリの整備により，UNIX関係のツールならそのままコンパイルして持ってくることが可能になり，プログラミング関係のツール類のほとんどはすでに自作する必要がなくなったといえる状況になっています。X68000がパソコンの枠を超えた開発環境を構築できるのもツール類の充実があればこそのことです。

microEMACS（またはnemacs）上でソースを作りそのままコンパイル，makeは効率のよいコンパイルを保証し，ファイルの保守にはRCS（リビジョンコントロールシステム）といったツールが活躍します。

そして，これらの要としてGCCの存在があります。

GCCを使う際に問題となるのが，GNU LIBの扱いです。これは68000ではサポートされない数値演算命令などをサポートするための簡単なライブラリですが，生成されるオブジェクトの中に必ずといっていいほど入り込みます。では，GCCでコンパイルされたオブジェクトはGNUの配布規定に従わなければならないのでしょうか。

GNU一般公有使用許諾書（GPL）や関連書籍を見る限りではこれはGNUウェアとして扱わなければならないように思われます。これらは法律的な解釈ですので，きわめて広範にわたっており，たとえば，GNU LIBに代わるものを作成して使用してもそれはGNUウェアとみなされることになります（らしい）。

対応としては，無条件に無視する，または，とりあえず無視する，しかたがないので無視する，といったところが一般的です。突き詰めればコンパイラの出力するオブジェクトにライセンスを認めるかという問題に突き当たるからです。

ちなみに，こういった問題は68030などでは発生しません。すべてコードとして生成されるからです。また，同じソースコードをXCでコンパイルした場合も問題は発生しません。

コンパイルされたオブジェクトに対してコンパイラの著作権を主張することはナンセンスです。GNULIBはコンパイルされたオブジェクトの不備を補うものであり，あくまで便宜上ライブラリの形式をとっているものともみなせます。

このあたりの解釈はどれが正しいといえるものではありませんが，馬鹿正直にGPLを読むと，オンラインソフト（市販ソフトも）の自由な流通を阻害することは明白です。それはおそらくGNUの基本精神に相反する結果を生みます。

ワークステーション環境とは違い，パーソナルコンピュータの世界でGNUの規定に従うとかえって制限が増えることもあります。もともとGCCはGNUに賛同する人しか使用できないものですから，GNUの規定云々によらず自主的にソースコードの公開などを進めていくべきでしょう。しかし，ソースコードもなしに「配布条件はGNUに準ずる」などとしたオンラインソフトも見かけますが，GCCの普及に比べGNU自体が一般には理解されていない気もします。

最適化の限界

GCCは，全体的に見るとへたなアセンブラプログラマが記述するよりもはるかに効率のよいコードを作成します。

それではC言語で作ったプログラムはなぜ遅いのでしょうか。

これはライブラリによる部分が足を引っ張っていることが考えられます。ライブラリのコードの質はともかく，どんな優秀なコンパイラでも，最適化が行われるのはユーザープログラムの内部だけです。

たとえば，ライブラリ側ではある計算結果をD0レジスタとD1レジスタに得たとしましょう。関数では2つの値を返すことはできませんから，それらをメモリ上に並べて置き，その先頭アドレスを返すことになります。メインプログラムではそれをまたD0とD1に読み込んで使っている……などという状況も往々にしてありうるわけです。

通常の関数呼び出しでコンパイラがどんなに無駄な処理を省いたとしても，最適化の及ばないライブラリとのインタフェイスが無駄を含んでしまいます。最高の品質のライブラリとコンパイラオブジェクトでも避けられない問題といえます。

ということなら，ライブラリを用途別に細分してインライン展開すればなかなか凄そうな気もします（一部ではできるらしいが）。しかし，GCCが中級アセンブラプログラマ以上のコードを出せるのであれば，最終的な性能としても，その程度のものを期待することは当たり前ではないでしょうか。

初心者のためのポインタ解説

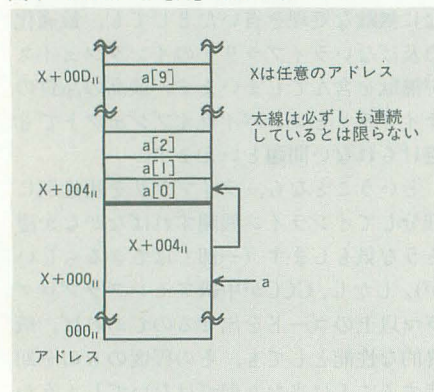
とりあえずポインタを制す

Kikuchi Isao 菊地 功

C言語入門者にとって最初の大きな難関といえば，やはりポインタでしょう。ここではコマンドラインパラメータの処理を題材にしてポインタの基本的な考え方を見ていってみましょう。

大枚をはたいてCコンパイラを買い，腕のつる思いをして持って帰ったのに，BAStoCしか使わないという方も多いかと思います。しかし，そんなあなたはコンバートされたCのソースを眺めたことがありますか？ X-BASICはそれ自体Cをかなり意識した(コンバートすることを前提とした)言語であり，X-BASICを理解できる方であれば，Cはそう手強い言語ではありませんし，ソースをある程度読み下すことができるでしょう。ただし，多くのCの入門者が戦いを挑んで敗れ去ったポインタや構造体などは避けては通れない問題です。そこで今回はポインタの攻略法を説明していきます。

基礎知識

パソコンに限らず，コンピュータとは大きく分けて考える部分と記憶する部分からなっています(入出力なんてのもありますが)。考える部分というのはCPU(中央処理装置)などであり，X68000ではモトローラという会社が開発したMC68000というLSI(と同等品)が使われています。頭脳に当たるだけに，スペックを語るときには欠かせないものです。それに対して，記憶する部分というのはメモリとかRAMとか呼ばれるコンデンサの塊です。

CPUはこのメモリに逐一事柄を記憶し，それを参照して行動しているのです。また，CPUの行動規範もメモリに書かれており，CPUは書かれているとおりにしか行動できません。これがプログラムと呼ばれるものです。

さて，CPUはメモリに書かれている命令に沿って行動しますが，自分がいまどこを実行しているかわからなくならないのでしょうか？ たとえば，メモ用紙に命令が羅列してあるだけでは，どこまで実行したのかわからなくなって同じことを繰り返してみたり，命令を飛ばしてしまったりします。

そこで，ちゃんと罫線の引かれたノートにページと行番号をふり，そこに命令を書くようにします。そうすればいま実行している命令のページと行番号さえ覚えておけば迷う心配はありません。メモリ上でこのページと行番号というものに相当するのがアドレスと呼ばれるものです。読んで字の如く，メモリの「住所」を示しており，一般にアドレスの低いほうから順に使われていきます。

また，変数などのデータもアドレスに従って格納されており，Aという変数を参照する場合，実際のプログラムではAが格納されているアドレスから値を拾ってくるという操作が行われています。つまりAという変数名はプログラムを作成する段階において人間にわかりやすくするためのものであって，実行の段階においてはインタプリタなりコンパイラなりがすべてアドレスに置き換えてしまうのです。そこをきっちり理解できていれば，ポインタなんてどってこた～ありません。

配列変数

まず，本題に入る前に，C言語での配列について説明しましょう。配列というのはよく箱（変数）がたくさんくっついたものという説明をされます。X-BASICでは，

```
dim char a(10)
```

などのように宣言されますが，C言語では，

```
char a[10];
```

のように記述されます。括弧がちょっと変わっただけであとは同じようなものです。

ただ，注意しなければならないのは，上のように宣言した場合，BASICではa(0)～a(10)までの配列を使用できますが，Cではa[0]～a[9]までの10個の配列を宣言したことになります。これは言語の仕様ですので，覚えるよりしかたがありません。

さて，配列変数の利点はなんでしょう。a0～a9までの変数を宣言した場合となにが違うのでしょう。簡単にいってしまえば，添字に変数を使えることです。たとえば，上の10個の変数に1から9までの値を入れるとしましょう。a0～a9の変数を宣言した場合は，代入式を10個並べる必要があります。しかし，配列変数であれば，

```
for( i=0; i<10; i++ ) a[i]= i;
```

と1行で書くことができます。

この程度であればちょっとした手間ですみますが，もっと複雑になった場合，配列の数だけ場合分けするなどといったことが不要になります。

では，この配列変数は実際どのように管理されているのでしょうか？ 普通の変数はコンパイルされるとき（正確には実行ファイルがメモリにロードされるとき）にアドレスに変換されるといいました。ところが配列変数は，添字に変数を用いることができるため，実行してみないとその変数がどのアドレスに対応するのかわかりません。a[i]の内容がどこにあるかは，iの値によって変わってくるのは当然のことです。

そこで，配列変数は先頭のa[0]のアドレスだけを覚えておいて，a[i]を参照するときには，先頭のアドレスからi番目といった具合に，必要に応じてアドレスを計算する

図1 char a [10] のメモリマップ

方法をとっています。ですから，配列変数を使用した場合は若干速度が落ちますが，よほど大量の配列をアクセスしない限り，差はでないでしょう。

ポインタ

配列変数がわかったところで，いよいよポインタに移りましょう。といっても，実は配列変数もポインタも基本的に似たもので，宣言時にメモリを確保するかしないかという違いしかありません。ポインタとは「指し示すもの」という意味ですが，その名のとおりアドレスを指しているにすぎません。ポインタは，

```
char *a;
```

のように宣言しますが，この場合はaという32ビット型変数（アドレスは32ビット）を確保し，その変数が8ビットデータが格納されているアドレスを示すポインタであることを'*'でコンパイラに知らせているようなものです。

たとえば，E00000$_H$というアドレスにアクセスしたい場合，

```
a = (char *)0xE00000;
```

と初期化し，なにか値を入れたければ，

```
*a = 100;
```

などとするわけです。この'*'は間接演算子などと呼ばれます。

また，それとは反対にアドレス演算子と呼ばれる'&'というものもあります。たとえば，iという変数に対して，&iはその変数iが格納されているアドレスを示します。

さて，先ほど配列とポインタは似たものだといいました。どこが似ているのでしょう。実は，

```
char a [10]
```

として宣言された配列変数は，ポインタ*aで参照することができ，その反対もまた可能なのです。配列で宣言したものは，先頭アドレスがころころ変わると厄介ですのでポインタで参照することはあまりありませんが，VRAMを参照する場合などは，

```
unsigned short *vp;
```

として宣言し，

```
vp = (unsigned short *)0xC00000;
```

としてVRAMの先頭アドレスを指しておいて，(x,y)座標のピクセルをvp [x+y*512]（1ライン512ワード）のようにすることがたびたびあります。

ここで問題になってくるのは，「じゃあ実際はvp[x+y*512]のアドレスはいくつなんだ」ということでしょう。C00000$_H$を先頭にx+y*512番目だからC00000$_H$+x+y*512だろう，というのは間違いです。vpはunsigned short(16ビット)で宣言されているので，C00000$_H$+(x+y*512)*2が正解です(アドレスの単位は8ビット)。このようにアドレスは宣言された型に従って演算が行われます。

では，vpに1を足してみたらどうなるでしょう。vp = vp+1としても結構ですし，1を加えるだけならC言語ではvp++という記述もできます。C00000$_H$に1を加えるから……。これは実はC00002$_H$になるのです。一見理不尽に感じるかもしれませんが，このほうが使うには都合がいいでしょう。もしなんらかの理由でC00001$_H$からアクセスしたい場合は，改めてアドレスを指定し直さなければなりません（もっとも，奇数アドレスからワードアクセスすればアドレスエラーになりますが)。

「アドレス」というものを理解できればさほど難しくもないでしょう。安藤さんに手紙を届けるのに，「安藤さん家」（安藤さんはひとりしかいないとする）と「新宿三丁目四番五号」という2通りの指定のしかたがあるのと同じようなものです。

多次元配列

いままでは一次元配列の話をしてきましたが，配列は一次元に限ったものではありません。X-BASICで，

```
dim char a(10,20)
```

のような二次元配列はC言語では，

```
char a[10][20];
```

のように宣言することができます。また，一次元のとき同様，ポインタでも**aと間接演算子を2つ並べることで，同じ働きをしますし，三次元以上でも同様です。では，このような場合の内部での管理はどのようになっているのでしょう。

一次元のときを思い出してください。ポインタとは，いってみればあるデータへのアドレスが格納されている変数のようなものでした。そして，そのポインタ変数自体，32ビットのメモリ上のデータですので，それにもアドレスがつけられています。

たとえば，a[5][8]にアクセスするには，まずaの値をアドレスの先頭として5×20+8番目先の1バイトに対して操作を行うという手順を踏んでいるわけです。三次元以上も同様ですが，多少複雑ですので，各自よく考えて理解してください。

図2 char a [10] [20] のメモリマップ

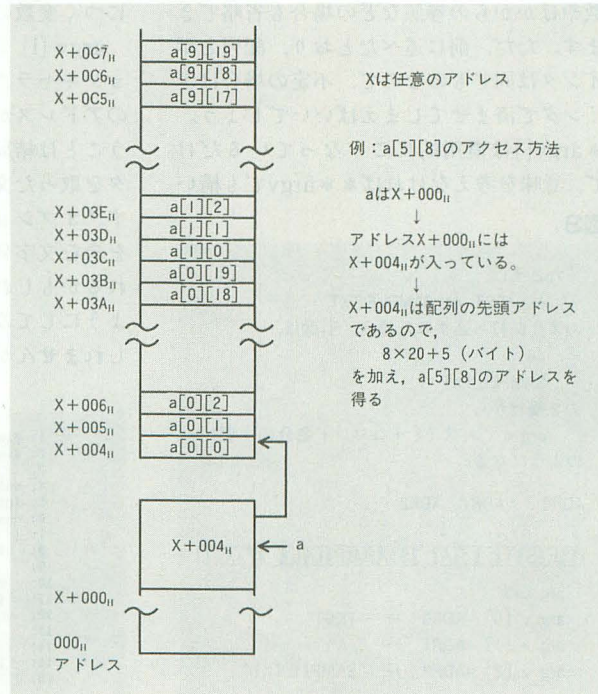
応用

理屈を並べ立てるだけではよく理解できないかもしれませんので，実際のプログラムに沿って実習してみましょう。

BASICをただひたすらコンパイルしていた方は，コマンドラインからの引数をどうやって受け取るか悩んでおられるのではないでしょうか。引数とはコマンド実行時にコマンド名に続けて入力してやる情報で，ファイル名やオプションなどがあります。このようなものを拾うには，残念ながらBASICだけではできませんので，C言語にコンバートしてから手を加える必要があります。

C言語では，まず最初に処理が移されるユーザー関数はmain()という名前に決められています。この関数は，BASICをコンバートしたC言語のソースを見たことのある方ならわかるかもしれませんが，

```
void main( argc, argv )
int   argc;
char  *argv [] ;
{
  :
```

といった具合に書き出されていると思いま

す（実際はb_argcとb_argvだけど，変数名の違いだけです）。これはmain()関数は２つの引数を取り，その下の２行はそれぞれの引数の型を示しています。

このargcとargvというのがコマンドラインからの引数の情報で，argcが引数の数，argvが引数へのポインタを示しています。argvは二次元配列になっていて，一次はそれぞれの引数の先頭のアドレス，二次は文字列を示しています。

言い忘れましたが，BASICでは文字列はstr型の変数を用いましたが，C言語ではキャラクタデータの集まりとして，char型の一次元配列で表されます。ですから，argv[i] [j] には，i番目の引数のj文字目のキャラクタコードが入っていることになります(argc<iの場合)。ただし，注意しなければならないのは，自分自身のコマンド名も引数とみなされていることです。たとえば，getargというコマンドを実行する場合，

getarg /?

とすると，argcには２，argv [0] には“getarg”という文字列へのポインタ，argv [1] には“/?”という文字列へのポインタが入ることになります。

さて，だいたいのことはおわかりいただけたと思いますが，argvの型宣言で「おや？」と思われたかもしれません。“[”と“]”の間に大きさが書かれていません。ここには引数の数が入らなければならないのですが，引数の数は実行してみなければわからないので，不定という意味で省略が許されています。また，argvだけでなく，引数やほかからの参照などの場合も省略できます。ただ，前に述べたとおり，配列とポインタは同じものなので，不定の場合はポインタで済ませてしまえばいいでしょう。＊argv[]は慣用的にこうなっているだけで，意味を考えなければ＊＊argvでも構いません。

リスト１を見てください。これは引数を拾ってただ表示するだけのプログラムです(引数はオプションとファイル名のみであると仮定します)。単体では意味を成しませんが，これを参考にして引数の構造を理解してください。

“#include”についてはほかの専門書に譲るとして，４～６行目は先ほど説明したとおりです。まず，argv [0] には必ずコマンド名が入りますので，“command:”として表示していますが，表示にはprintf()関数を使っています。詳しい説明はCライブラリマニュアルに譲りますが，ここではダブルクォーテーションの中の“%s”の部分がargv [0] の指し示す文字列に変換されて表示されると思ってください。

次にfor制御文でiを１からargc-1までループさせます。これは，iをargvの指数に使うためです。argv [0] はすでに表示しましたし，argvがargc個の配列ならば使用できるのはargv [0] ～argv [argc-1] までですよね？　次にオプションかファイル名かを判断するのですが，ここでは引数の先頭がオプションキャラクタかどうかで判断してあります。Human68kではオプションキャラクタの標準は'/'ですが，'-'も準標準となりつつありますので，両方に対応させます。

その後はそれぞれ“option:”，“filename:”に続けて表示させるのですが，13行目のオプションの表示はさっきとはちょっと違うようです。'&'は以前に少しお話ししましたね。これはアドレス演算子といって，後ろにつく変数のアドレスを示すものでした。

argv [i] [1] っていうのはつまりオプションキャラクタの次の文字ですよね。そこのアドレスが指し示す文字列を表示，ということは結局は先頭のオプションキャラクタを取った文字列を表示することになります。「オプションとはオプションキャラクタを含む文字列をいうんだ」という方もおられるかもしれませんが，学習のためにこのようにしてみました。ちょっと難しいかもしれませんが，ぜひ身につけてください。

打ち込んだらコンパイルです。テキストエディタで仮にリスト１をgetarg.cという名で入力・保存したとして，Cコンパイラをインストールして環境変数includeとlibを設定してあれば，コマンドラインから，

cc getarg.c

と入力するだけです。入力ミスがなければgetarg.xという実行ファイルができているはずです。もし，コンパイラが途中で止まってしまったら，リスト１とよく見比べて，間違いを修正したあとに再びコンパイルし直してください。

また，gccでもコンパイルできますので，持っておられる方はそちらのドキュメントなどを参考にしてコンパイルしてみてください。　コンパイルができたら実行してみましょう。“getarg”に続けてなにか入力してみてください。

```
A>getarg /opt1 -opt2 fn
command: A:¥getarg.x
option:     opt1
option:     opt2
filename:   fn
```

若干の違いがあるにせよ，だいたい上のようになったはずです。もし，全然ちがうようでしたら，もう一度リストを見直してみてください。

＊　　＊　　＊

C言語のポインタを中心に話をしてきましたが，理解できたでしょうか？　実際は若干異なるかもしれませんが，これまで説明してきたように理解していただいて問題ないと思います。ここまで書いて，私は無責任ながら「こんなんで理解できれば苦労しねぇよなぁ」と思ってたりします。いいわけではありませんが，確かにこの程度で完全に理解するのは無理でしょう（私の説明が悪いのも確かですが）。そもそもこういったことは，自分でプログラムを書いていくうちに少しずつ身についていくものです。今回の記事で，私は皆さんにそのための道しるべは示すことができたと思います。あとは皆さん次第ですので，ぜひ恐れずに突き進んでいってください。

図3

```
たとえば,
  A>TEST /A SAMPLE.TXT
のように打ち込まれた場合，引数は，
  /A
  SAMPLE.TXT
の２個だから，
  arg_c  = 3（2＋コマンド名分の１個）
のようになる。

ADR0     ADR1  ADR2
↓        ↓     ↓
|T|E|S|T| |/|A| |S|A|M|P|L|E|.|T|X|T|

 arg_c=3
 arg_v [0] =ADR0  →  "TEST"
 arg_v [1] =ADR1  →  "/A"
 arg_v [2] =ADR2  →  "SAMPLE.TXT"
```

リスト1

```
 1: #include  <stdlib.h>
 2: #include  <stdio.h>
 3:
 4: void      main( argc, argv )
 5: int       argc;             /* 引き数の数 */
 6: char    - *argv[];          /* 各引き数へのポインタ */
 7: {
 8:   int     i;
 9:
10:   printf( "command: %s¥n", argv[0] );                   /* コマンド名 */
11:   for( i=1; i<argc; i++ ){                              /* 引き数の数だけ繰り返す*/
12:           if( argv[i][0]=='-' || argv[i][0]=='/' )      /* 引き数の先頭が'-'か'/'*/
13:                   printf( "option:  %s¥n", &argv[i][1] ); /* オプションとみなす */
14:           else    printf( "filename:%s¥n", argv[i] );   /* それ以外（ファイル名）*/
15:   }
16: }
```

基礎的なファイル処理

C言語によるデータ処理入門

Tan Akihiko 丹　明彦

初心者が悩みがちなのは，なにをプログラムするかです。まずは身近なものから始めていくことでしょう。ここでは用途の広いファイル処理を題材に，基本的Cライブラリの使い方に馴染んでください。

そろそろ単車をいまのものに乗り換えてから1年半が経過し，そろそろ借金生活にも別れを告げることができそうです。それを記念して，というわけではありませんが，1年半の間こつこつとためてきたガソリンスタンドの領収書をネタにデータ処理の真似事でもやってみたいと思います。

私が自動二輪の免許を取ったのはもう4年半前のことになります(いまだに普通免許は持っていません)。最初の単車は中古だったのですが，いまのものは新車で買ったので，ひとつマメに記録をつけてみようと思ったのです。ガソリンスタンドで給油するたびに，領収書の余白に走行距離を書きつけていき，捨てずに取っておきました。

1年半で8600km。少ないですねえ。週末くらいしか乗りませんからねえ。領収書は40枚くらいで，データ処理としての格好がつくくらいにはなっているでしょう。

これから新車を買おうという方は，ひとつこうした記録を取り続けて，何年かのちに計算機で処理してみてはいかがでしょうか。自分の愛車や自分の運転について，なにかが見えてくるかもしれません。

ちなみに私の場合，燃費が知りたくてこの記録を取ってきたのですが，今回計算してみて，この車種の一般的な値に比べるとどうも燃費が悪いことがわかりました。あまりツーリングに行かず，東京都内の渋滞路を走ることが多いという行動様式のせいもあるのかもしれませんが，根本的に乗り方が悪いという気が最近しています。ここはひとつ，再出発という気分で，乗り方を変えていこうと考えています。

＊　＊　＊

というわけで，ちょいと地味な気もしますが，今回は，

データファイルから情報を読み取り，
その情報を加工して別の情報を得て，
結果を出力する

というプログラムをC言語を用いて作ってみることにします。このプログラムを作る過程では，

テキスト形式のファイルの読み書き
ある情報から別の情報を計算する方法
数値情報を視覚化する方法

を使うことになります。ついでに，

分割コンパイルのための変数や関数の構成のしかた
メイクファイルの書き方

といった，Cとの上手なつきあい方に関するノウハウも出てきます。メイクファイルの書き方については別稿で解説します。

今回はポインタとか配列とか構造体とかいったC入門者にとっての「定番」ハードルについては深入りしません。

データ形式について

読み込みを楽にしたいので，ある程度フォーマットを固定します。まずデータを行単位とします。ファイルに記述する行には3種類あります。データ行とコメント行，それに終了行です。

データ行はいちばん大事です。1行に1回の給油ぶんのデータ，つまり年，月，日，走行距離，給油量，ついでに値段(税別)を書きます。これらのデータをフィールドと呼ぶことがあります。

各フィールドはスペースまたはタブで区切られています。これを，「セパレータは空白文字である」といいます。もっとメジャーなセパレータとしてはコンマ「,」が考えられますが，私はCの標準入出力ライブラリで楽をしたいと考えているので，セパレータに空白文字を選びました。

コメント行は記号「#」で始まる行(1カラム目が「#」である行)です。その右側にはなにを書いてもかまいません。プログラム側で無視します。

そして終了行は先頭の文字列が「END」である行です。

一般にこの手の情報は個数が不定であり，その個数を表す手段が必要になります。今回の場合は，給油の回数(領収書の枚数)が決まっていません。40回というのは，たまたま私がそうだったというだけのことです。

「データ行の中身」のように個数が決まっているものの読み込みは楽です。しかし，「データ行の行数」のように個数が決まっていないものにはそれなりの対応が必要です。やり方は主に2つあります。

ひとつは，データファイルの最初にデータがいくつあるか書いておく方法です(図1)。これはプログラマにとっては楽なのですが(なにしろforループ一発ですみますから)，データファイルを書くほうにとってはあまり嬉しいものではありません。データの個数を数えなくてはならないというのが主な理由です。

また，データを追加した場合は数え直しが必要になり面倒で，間違いのもとにもなります(図2)。「本当のデータの個数(ファイルに実際に記述されているデータの個数)」と「ファイルの先頭に記述するデータの個数」という，等しくなくてはならない

図1　個数を最初に記述する

```
# 年  月  日       km     ℓ     円
5
1991  12  06     13.0  13.9  1904
1991  12  20    258.0  13.1  1795
1992   1  24    499.0  13.0  1742
1992   2  29    767.0  14.0  1876
1992   4  25    944.6  11.4  1528
```

図2　データを追加すると最初の行も変更する必要がある(間違いのもと)

```
# 年  月  日       km     ℓ     円
6
1991  12  06     13.0  13.9  1904
1991  12  20    258.0  13.1  1795
1992   1  24    499.0  13.0  1742
1992   2  29    767.0  14.0  1876
1992   4  25    944.6  11.4  1528
1992   5  22   1144.3  12.4  1662
```

2つの量が2カ所に存在しているというのが本質的な問題です。これらが一致しなければ，データは間違ったものとなります。

もうひとつは，データファイルの終端を表現するための記号を決めておく方法です(図3)。これだと，データの個数をプログラム側で数えてやる必要があるのですが，こうして得たデータの個数は常に本当のデータの個数と一致します。特に今回のように，あとから追加していくことの多い種類のデータでは，追加する行以外に修正する行がないため，こちらのやり方が適しているといえましょう(図4)。

終端の表現としては，今回のように「END」などの終端記号を使うもの，またはもっと徹底してファイルそのものの終端を検出する方法(Cのライブラリではfeof()などを使います)があります。

＊　＊　＊

今回は，入力データファイルが「紳士的に」書かれていることを期待して，読み込み時のエラーチェックを一切行っていません。たとえば，終端記号「END」を忘れただけでも悲惨なことになるでしょう。

ライブラリ関数のご利益

こういってはなんですが，Cは「なんにもない」プログラミング言語といえます。特に入出力関係は気持ちいいほどなんにもありません。BASICでいうところのINPUT文とかPRINT文とかいうものはCの言語仕様にすら入っていないのです。けれども誰もそうは思っていないことでしょう。

その理由のひとつには，そうした入出力そのものをCで書けるということがあります。たとえばPRINT文に相当するものはDOSコールなどとのインタフェイスを上手に使って実現することになります。まあこれは半分は思い込みのようなものでして，Cの生まれ育った当時のUNIXならともかく，OSが出来あいのものであること(ソースプログラムが付属しないこと)が当たり前のように受け入れられているHuman68kや近年のUNIXの世界では事実上成り立たないものであります。ただ，Cは依然としてそれができる言語であるというのは事実です。

もうひとつには，こちらが本題なのですが，まっとうなCの処理系ではライブラリが充実しているためです。たとえばPRINT文に相当する機能はstdio(スタンダードI/O:標準入出力)ライブラリに含まれるprintf()などのライブラリ関数によって実現されています。基本的にこうしたライブラリはCの言語仕様とは独立ですが，近頃はライブラリそのものの仕様を定めた標準規格もいくつかできています。

図3　終端記号を決める

# 年	月	日	km	ℓ	円
1991	12	06	13.0	13.9	1904
1991	12	20	258.0	13.1	1795
1992	1	24	499.0	13.0	1742
1992	2	29	767.0	14.0	1876
1992	4	25	944.6	11.4	1528
END					

図4　追加してもほかの行に影響しない

# 年	月	日	km	ℓ	円
1991	12	06	13.0	13.9	1904
1991	12	20	258.0	13.1	1795
1992	1	24	499.0	13.0	1742
1992	2	29	767.0	14.0	1876
1992	4	25	944.6	11.4	1528
1992	5	22	1144.3	12.4	1662
END					

ライブラリ関数を使ったデータ入力

今回扱うのは先ほど紹介したようなフォーマットのついたテキストファイルです。

まずは入力からいってみましょう。用いるライブラリ関数はfopen()，fgets()，sscanf()，fclose()の4つです。いずれも，ヘッダファイルstdio.hをインクルードしておいて使います。インクルードするには，プログラムの最初のほうで，

```
#include <stdio.h>
```

とします。これはスタンダードI/Oライブラリを使うときの約束のようなものです。それでは各関数の説明にいきましょう。

●FILE＊fopen(ファイル名，モード)

ファイルをオープンします。読み出しモードと書き込みモードを選ぶことができます。ここではファイルの入力を行いたいのですから，読み出しモードでオープンします。Human68kでは，ファイルにはテキストファイルとバイナリファイルの区別がありますので，モードにはそれを示す修飾子をつけます。読み出しモードの"r"，テキストモードの"t"，でモードには"rt"を指定するのが一般的でしょう。

```
例：FILE＊fp;
    fp=fopen("a:¥config.sys","rt");
```

戻り値としてファイル情報を収めた構造体へのポインタ(ファイルポインタと呼ぶことにします)を返します。このとき，構造体の中身について知っておく必要はありません。ファイルポインタを宣言しておいて，戻り値をそれで受けてください。

今後，このファイルポインタはファイルのアクセスに使います。なお，読み込みモードの場合は，読み込むべきファイルが存在しない場合などにファイルのオープンに失敗しますが，このとき戻り値のファイルポインタにはNULL(実際の値は0)が入って返ってきます。ファイル名の指定間違いなどでオープンに失敗することはよくあることです(特にコマンドラインの引数をファイル名として使う場合など)ので，面倒でもチェックは必ず行うようにしてください。ここをさぼると，あとあとバスエラーやアドレスエラーに悩まされることになります。

●char＊fgets(バッファ，文字数，ファイルポインタ)

ファイルポインタで示されたファイルから，1行をバッファに読み込みます。ファイルポインタは先ほどのfopen()で得た値を使います。バッファは各自で十分な長さのものを確保してください。通常のテキストファイルなら，画面の横幅，X68000なら96文字，安全を期して128文字もとっておけばいいでしょう。文字数には，バッファのサイズを指定します。バッファのサイズをオーバーして読み込むことを防ぐために指定すると考えておいてください。

戻り値は，読み込みに成功すればバッファのアドレスが，失敗すればNULLが返ってきます。エラーチェックをするのが望ましいでしょう(今回はさぼりました)。

●int sscanf(バッファ，書式文字列へのポインタ，読み込み先変数へのポインタ……)

バッファ(fgets()で読み込んだものを使います)から，書式文字列に示された形式でデータを読み出し，ポインタで示された変数に代入します。と言葉で書くとわかりづらいですが，使い方も最初のうちはわかりづらいものです。特にリファレンスマニュアルの記述には圧倒されることでしょう。ここは，よく使うものについて形を覚えるのが得策といえます。

```
例：FILE＊fp;
    char buffer [128];
    int i;
    double d;
    char s [32];
    fopen("a:¥test.dat","rt");)
    fgets(buffer,128,fp);
    sscanf(buffer,"%d",&i);
    fgets(buffer,128,fp);
    sscanf(buffer,"%lf",&d);
    fgets(buffer,128,fp);
    sscanf(buffer,"%s",s);
```

上の例では，sscanf()で始まる行が3つあります。それぞれ，整数，倍精度実数，文字列（Cでは文字型整数の配列ですが）をファイルから読み取るための関数呼び出しです。ちょっと観察すればわかりますが，書式指定文字列は"%"で始まり，変数の型を示す文字で終わります。特に注意してもらいたいのが2番目の倍精度実数で，

sscanf(buffer,"%f",&d);

のように，"l"(longを意味するのでしょうか？)をつけ忘れるとまったくうまくいきません("%f"は単精度実数)。恥ずかしながら，私は今回これではまりかけました。

書式指定文字列にはひとつだけでなく複数のフィールドも記述できます。

sscanf(buffer,"%d %lf %s",&i,&d,s);

という書き方もできるのです。

戻り値は読み込んだフィールドの数またはエラーを示す値ですが，私は特に気にしていません。

●int fclose(ファイルポインタ)

指定したファイルをクローズします。あと始末はきちんとしましょう。戻り値は正常終了したかどうかですが，これも特に気にしていません。

ライブラリ関数を使ったデータ出力

次はテキストを出力する方法を紹介します。用いる関数はprintf()です。これもヘッダファイルstdio.hをインクルードしておいて使います。

●int printf(書式文字列へのポインタ，値……)

標準出力（通常は画面）に，値を書式文字列に従ったフォーマットで出力します。printf()の書式指定はけっこう強力で，数値の桁を揃えて出力するといったことが簡単にできます。sscanf()と同様，リファレンスマニュアルの記述はけっこうわかりづらいので，これまた形から入るとよいでしょう。

例：printf("%4d¥n",i);
printf("%6.1f¥n",d);
printf("%24s¥n",s);

上から順に，4桁のフィールドに右詰めで整数を，6桁のフィールドに実数を小数点以下1桁だけ，24桁のフィールドに文字列を，それぞれ出力します。最後の"¥n"は改行文字です。

書式指定文字列は複数書くことができ，さらに書式指定と関係のない文字を書いておけばそのまま出力されます。たとえば，

printf("データ%4d: %6.1f(名称%5s)¥n",i,d,s);

のような場合，次のように出力されます。

データ 312： 3487.2(名称stdio)

戻り値は出力した文字数ですが，気にしなくてもいいでしょう。

燃費計算プログラムへの応用

以上の知識を使って，私の1年半の蓄積を処理するプログラムを書いてみました。プログラムは2つで，ひとつはテキストベースのnenpi_t.x，もうひとつはグラフィックベースのnenpi_g.xです。XC ver.2とgccの両方でコンパイルして動作を確認してあります。

ライブラリ関数の使い方は慣れればすむ話ですから，ここでは元データをどのように加工したかについて解説しましょう。

元データに記録したのは，各給油時の日付と総走行距離，給油量と値段です。

主に求めたいのは燃費です。これは前回の給油時から今回の給油時までに走った距離を今回の給油量で割れば求めることができます。

ただしこれは，毎回の給油で満タンにしているという条件の下で成立します。なぜなら，たとえば前回の給油で満タンにしなかったとすると，走行距離が必然的に短くなり，そのわりに給油量が多くなるので，燃費の値の著しい悪化を招きます。これを防ぐために，給油は満タンにするということを守りましょう。これとて，ガソリンスタンドによって，または日によって満タンの基準が変わりますから，必ずしも正確とはいえません。この手のことに誤差はつきものですから，長期間にわたって記録を取り，平均を求めるくらいの心構えが必要ですね。話がプログラミングから大きく逸れていますが，実験データを整理するという場面では大切なことです。

平均の燃費を求めるためにはもうひとひねり必要です。それは初回の給油の処理です。私のデータでは，13km走行時に13.9リットル給油したことになっています。当然のことながら，通常の燃費計算式を当てはめることはできません（1km/ℓを切ってしまいます）。これからわかるのは，初回は例外として扱う必要があるということです。

今回は次のようにしました。

1) 初回は燃費を計算しない。
2) 平均燃費を計算するための総走行距離は，各回の総走行距離から初回の総走行距離を引いたものとする。
3) 平均燃費を計算するための総給油量は，給油量の合計から初回の給油量を除いたものとする。
4) 平均燃費は2)を3)で割って求める。

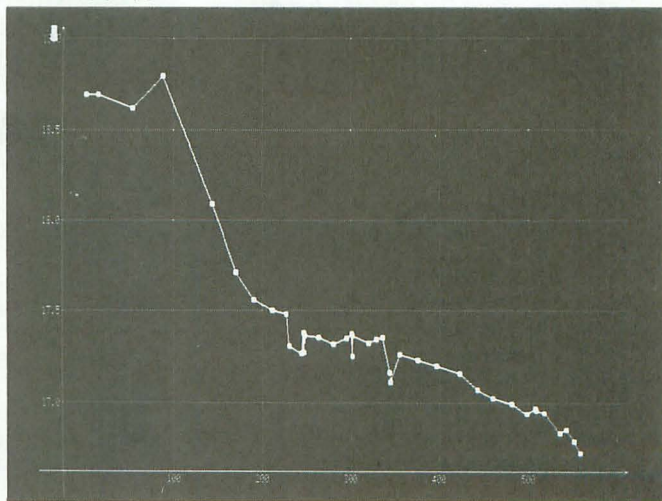
テキストベースのプログラムの流れは給油の記録を書いたファイルから読み込んだデータから燃費などを計算し，求めた数値をプリントするだけの簡単なものです。

グラフィックベースのプログラムは，データを読み込むところまでは共通ですが，出力がグラフになっています。平均燃費などの移り変わりが目に見えて面白いかもしれません。

プログラミング上はたいしたことをしていませんが，ひとついっておくと，グラフィック処理関数を自前で書いています。これは，最近ソフトバンクから出版されたパブリックドメインのCライブラリ「libc」（黒地に緑の字のもの）においてBASICライブラリがサポートされていないためです。ごくふつうのCプログラムなので，libcの環境はもちろん，XCの環境でも使えます。

おわりに

なんだか今回は，自分でもわかるくらいに小さくまとまってしまっています。プログラマになったからには，ぞくぞくするほど興奮させるプログラムに出合いたいし，そういうプログラムを書いていたいものです。今回のプログラムは，多少生活に関わりもあるし，具体的でわかりやすいもので

グラフの出力例

はありますが，決してエキサイティングなものとはいえないでしょう。

そもそも，こういう，お仕事マシンで表計算ソフトを使えばできてしまうようなことをわざわざCでやったというのが間違いなのかもしれません。が，私がときどき必要に駆られて作る短いプログラムのほとんどがこのタイプのもの，つまりファイルを開いてフォーマットされたデータを読み込むタイプのプログラムです。私は，お仕着せの作法を強要されている気がするためか，表計算ソフトがあまり好きではないのです。データの入力も加工も，あくまで手に馴染んだテキストエディタでやりたいのです。そんな私にとって，

fp = fopen(filename, "rt");

という文は，ほとんど考えずに書ける定型句のようなものです。いい換えれば，私はCとstdio.hで思考する人間なのかもしれません。

ただし，UNIX環境においては，awkなどのパターン走査プログラムがこの手の用途にうってつけであると申しあげておきましょう。X68000には，日本語化されたGNUのawkであるjgawkが移植されています。awkは，テキストファイル相手であれば，相当に複雑な処理もこなせます。

グラフ表示でやり残したこと

縦，横軸ともに，どう目盛りを切るかを自動的に決める機能がないというのはエレガントではありません。たとえば横軸は走行距離で，9000kmまでというのは読み込んだデータから明らかですから，その1/10くらいの値で適当に切りのいい数字，ここでは1000kmごとに目盛りを切ります。縦軸は燃費で，13km/l～20km/lあたりに分布していますから，0.5km/lごとに目盛りを切ります。これくらいの計算は自動でできるようにしておくべきでした。現状では，nenpig.cを見てもわかるとおり，ハードコーディングになっていますから，データファイルを替えた途端に破綻します。このあたりは算法を工夫すればけっこう面白い題材だとは思いますが……。

リスト1 nenpi_t.c

```
 1: /*
 2:  * nenpi_t.c
 3:  * - 燃費の統計処理みたいなもの(テキスト版)
 4:  * Jun. 1993            丹 明彦
 5:  */
 6:
 7: #include    <stdio.h>
 8: #include    "gasstat.h"
 9:
10: int         main( argc, argv )
11: int         argc;
12: char        *argv[];
13: {
14:     FILE        *fp;
15:     if ( argc != 2 ) {
16:             fprintf( stderr, "usage: %s filename¥n", argv[0] );
17:             return 1;
18:     }
19:     fp = fopen( argv[1], "rt" );
20:     if ( fp == NULL ) {
21:             fprintf( stderr, "%s: no such file.¥n", argv[0] );
22:             return 1;
23:     }
24:     readData( fp );
25:     fclose( fp );
26:     calcData();
27:     printData();
28:     return 0;
29: }
```

リスト2 nenpi_g.c

```
 1: /*
 2:  * nenpi_g.c
 3:  * - 燃費の統計処理みたいなもの(グラフィック版)
 4:  * Jun. 1993            丹 明彦
 5:  */
 6:
 7: #include    <stdio.h>
 8: #include    "graphlib.h"
 9: #include    "plot.h"
10: #include    "gasstat.h"
11:
12: int         main( argc, argv )
13: int         argc;
14: char        *argv[];
15: {
16:     FILE        *fp;
17:     int         i;
18:     double      d;
19:
20:     if ( argc != 2 ) {
21:             fprintf( stderr, "usage: %s filename¥n", argv[0] );
22:             return 1;
23:     }
24:     fp = fopen( argv[1], "rt" );
25:     if ( fp == NULL ) {
26:             fprintf( stderr, "%s: no such file.¥n", argv[0] );
27:             return 1;
28:     }
29:     readData( fp );
30:     fclose( fp );
31:     calcData();
32:     screen( 2, 0, 1, 1 );
33:     plotAxis( 14 );
34:
35:     for ( d = 17.0; d <= 19.0; d += 0.5 )
36:             plotNotchYDouble( averageNenpi, nRecharge, d, 14 );
37:
38:     for ( i = 100; i <= 500; i += 100 )
39:             plotNotchXInt( day, nRecharge, i, 14 );
40:     plotDataIntDouble( day, averageNenpi, nRecharge, 15 );
41: /*
42:     for ( i = 0; i <= 9000; i += 1000 )
43:             plotNotchXDouble( netDistance, nRecharge, i, 14 );
44:     plotDataDoubleDouble( netDistance, averageNenpi, nRecharge, 15 );
45: */
46:     return 0;
47: }
```

リスト3 gasstat.c

```
 1: /*
 2:  * gasstat.c
 3:  * - 燃費の統計処理みたいなもの
 4:  * Jun. 1993            丹 明彦
 5:  */
 6:
 7: #include    <stdio.h>
 8: #include    "graphlib.h"
 9:
10: #define     MAXRECHARGE     100
11:
12: int         nRecharge;
13:
14: /* ファイルに記述する量 */
15: int         year[ MAXRECHARGE ];                    /* 年 */
16: int         month[ MAXRECHARGE ];                   /* 月 */
17: int         date[ MAXRECHARGE ];                    /* 日 */
18: double      odoMeter[ MAXRECHARGE ];                /* 総走行距離 */
19: double      gasoline[ MAXRECHARGE ];                /* 給油量 */
20: int         money[ MAXRECHARGE ];                   /* 価格 */
21:
22: /* ファイルに記述しない(あとから算出する)量 */
23: double      tripMeter[ MAXRECHARGE ];               /* 走行距離 */
24: double      netDistance[ MAXRECHARGE ];             /* 純総走行距離 */
25: double      totalGasoline[ MAXRECHARGE ];           /* 総給油量 */
26: double      netTotalGasoline[ MAXRECHARGE ];        /* 純総給油量 */
27: double      nenpi[ MAXRECHARGE ];                   /* 燃費 */
28: double      averageNenpi[ MAXRECHARGE ];            /* 平均燃費 */
29: double      price[ MAXRECHARGE ];                   /* 1lあたりの値段 */
30: int         day[ MAXRECHARGE ];                     /* 使用日数 */
31:
32: void        readData( fp )
33: FILE        *fp;
34: {
35:     static char     tmp[128];
36:
37:     nRecharge = 0;
38:     for (;;) {
39:             /* 1行読み込む */
40:             fgets( tmp, 127, fp );
41:             /* 行頭が'#'であれば(コメント)次の行へ */
42:             if ( tmp[0] == '#' ) continue;
43:             /* "END"と記述した行なら終了 */
44:             if ( tmp[0] == 'E' ) break;
```

```
45:             /* 通常のデータと考えられるのでフォーマットにしたがって読み取る */
46:             sscanf( tmp, "%d %d %d %lf %lf %d",
47:                     &(year[ nRecharge ]),
48:                     &(month[ nRecharge ]),
49:                     &(date[ nRecharge ]),
50:                     &(odoMeter[ nRecharge ]),
51:                     &(gasoline[ nRecharge ]),
52:                     &(money) [ nRecharge ]);
53:             nRecharge++;
54:     }
55:     return;
56: }
57:
58: void        calcData()
59: {
60:     int     i;
61:
62:     tripMeter[0] = odoMeter[0];
63:     netDistance[0] = 0.0;               /* 初回給油時の総走行距離を初期値とする */
64:     totalGasoline[0] = gasoline[0];
65:     netTotalGasoline[0] = 0.0;          /* 初回給油状態を初期値とする */
66:     nenpi[0] = 0.0;                     /* 初回は燃費を統計に入れない */
67:     averageNenpi[0] = 0.0;              /* 初回は燃費を統計に入れない */
68:     price[0] = money[0]/gasoline[0];
69:     day[0] = 0;                         /* 初回給油日を初期値とする */
70:     for ( i = 1; i < nRecharge; i++ ) {
71:             tripMeter[i] = odoMeter[i] - odoMeter[i-1];
72:             netDistance[i] = odoMeter[i] - odoMeter[0];
73:             totalGasoline[i] = totalGasoline[i-1] + gasoline[i];
74:             netTotalGasoline[i] = totalGasoline[i] - totalGasoline[0];
75:             nenpi[i] = tripMeter[i] / gasoline[i];
76:             averageNenpi[i] = netDistance[i] / netTotalGasoline[i];
77:             price[i] = money[i] / gasoline[i];
78:             day[i] = (year[i]  - year[0])*365 +
79:                             (month[i] - month[0])*30 +
80:                             (date[i]  - date[0]); /* いいかげん */
81:     }
82:     nenpi[0] = nenpi[1];                            /* 初回の燃費にダミーの値 */
83:     averageNenpi[0] = averageNenpi[1];              /* 初回の平均燃費にダミーの値 */
84:     return;
85: }
86:
87: void        printData()
88: {
89:     int     i;
90:
91:     for ( i = 0; i < nRecharge; i++ ) {
92:             printf( "%d年%2d月%2d日(%3d日後) %5.1fkm/%6.1fkm %4.1fl %4.1fkm/l 平
均%4.1fkm/l %4d円 %5.1f円/l¥n",
93:                     (year[i]),
94:                     (month[i]),
95:                     (date[i]),
96:                     (day[i]),
97:                     (tripMeter[i]),
98:                     (odoMeter[i]),
99:                     (gasoline[i]),
100:                    (nenpi[i]),
101:                    (averageNenpi[i]),
102:                    (money[i]),
103:                    (price[i]) );
104:    }
105:    return;
106: }
```

リスト4 plot.c

```
  1: /*
  2:    plot.c
  3:    - グラフ描画
  4:    Jun. 1993         丹 明彦
  5: */
  6:
  7: #include    <stdio.h>
  8: #include    <string.h>
  9: #include    "graphlib.h"
 10:
 11: #define     WIDTH           768
 12: #define     HEIGHT          512
 13:
 14: #define     LEFTMARGIN      32
 15: #define     RIGHTMARGIN     32
 16: #define     TOPMARGIN       32
 17: #define     BOTTOMMARGIN    32
 18:
 19: #define     WIDTH1  (WIDTH-LEFTMARGIN-RIGHTMARGIN)
 20: #define     HEIGHT1 (HEIGHT-TOPMARGIN-BOTTOMMARGIN)
 21:
 22: void        plotAxis( p )
 23: int         p;
 24: {
 25:    /* x軸 */
 26:    line( 0, HEIGHT-BOTTOMMARGIN, WIDTH-1, HEIGHT-BOTTOMMARGIN, p, 0xFFFF );
 27:    line( WIDTH-1, HEIGHT-BOTTOMMARGIN, WIDTH-11, HEIGHT-BOTTOMMARGIN-4, p,
0xFFFF );
 28:    line( WIDTH-1, HEIGHT-BOTTOMMARGIN, WIDTH-11, HEIGHT-BOTTOMMARGIN+4, p,
0xFFFF );
 29:    /* y軸 */
 30:    line( LEFTMARGIN, 0, LEFTMARGIN, HEIGHT-1, p, 0xFFFF );
 31:    line( LEFTMARGIN, 0, LEFTMARGIN-4, 10, p, 0xFFFF );
 32:    line( LEFTMARGIN, 0, LEFTMARGIN+4, 10, p, 0xFFFF );
 33:    return;
 34: }
 35:
 36: void        getMinMaxInt( x, n, min, max )
 37: int         x[], n, *min, *max;
 38: {
 39:    int     i, dx;
 40:    *min = *max = x[0];
 41:    for ( i = 1; i < n; i++ ) {
 42:            if ( *min > x[i] ) *min = x[i];
 43:            if ( *max < x[i] ) *max = x[i];
 44:    }
 45:    /* 10%余裕を取る */
 46:    dx = *max - *min;
 47:    *max += dx/20;
 48:    *min -= dx/20;
 49:    return;
 50: }
 51:
 52: void        getMinMaxDouble( x, n, min, max )
 53: int         n;
 54: double      x[], *min, *max;
 55: {
 56:    int     i;
 57:    double  dx;
 58:    *min = *max = x[0];
 59:    for ( i = 1; i < n; i++ ) {
 60:            if ( *min > x[i] ) *min = x[i];
 61:            if ( *max < x[i] ) *max = x[i];
 62:    }
 63:    /* 10%余裕を取る */
 64:    dx = *max - *min;
 65:    *max += dx/20.0;
 66:    *min -= dx/20.0;
 67:    return;
 68: }
 69:
 70: void        plotNotchXInt( x, n, x1, p )
 71: int         x[], n, x1, p;
 72: {
 73:    int     xmin, xmax, px, w;
 74:    char    tmp[16];
 75:
 76:    getMinMaxInt( x, n, &xmin, &xmax );
 77:    px = LEFTMARGIN + WIDTH1 * (x1 - xmin) / (xmax - xmin);
 78:    line( px, 0, px, HEIGHT-BOTTOMMARGIN, p, 0x5555 );
 79:    sprintf( tmp, "%d", x1 );
 80:    w = strlen( tmp );
 81:    symbol( px-w*3, HEIGHT-BOTTOMMARGIN+2, tmp, 1, 1, 0, p, 0 );
 82:    return;
 83: }
 84:
 85: void        plotNotchXDouble( x, n, x1, p )
 86: int         n, p;
 87: double      x[], x1;
 88: {
 89:    int     px, w;
 90:    double  xmin, xmax;
 91:    char    tmp[16];
 92:
 93:    getMinMaxDouble( x, n, &xmin, &xmax );
 94:    px = LEFTMARGIN + WIDTH1 * (x1 - xmin) / (xmax - xmin);
 95:    line( px, 0, px, HEIGHT-BOTTOMMARGIN, p, 0x5555 );
 96:    sprintf( tmp, "%.1f", x1 );
 97:    w = strlen( tmp );
 98:    symbol( px-w*3, HEIGHT-BOTTOMMARGIN+2, tmp, 1, 1, 0, p, 0 );
 99:    return;
100: }
101:
102: void       plotNotchYInt( y, n, y1, p )
103: int        y[], n, y1, p;
104: {
105:    int     ymin, ymax, py, w;
106:    char    tmp[16];
107:
108:    getMinMaxInt( y, n, &ymin, &ymax );
109:    py = HEIGHT - BOTTOMMARGIN - HEIGHT1 * (y1 - ymin) / (ymax - ymin);
110:    line( LEFTMARGIN, py, WIDTH-1, py, p, 0x5555 );
111:    sprintf( tmp, "%d", y1 );
112:    w = strlen( tmp );
113:    symbol( LEFTMARGIN-2-w*6, py-6, tmp, 1, 1, 0, p, 0 );
114:    return;
115: }
116:
117: void       plotNotchYDouble( y, n, y1, p )
118: int        n, p;
119: double     y[], y1;
120: {
121:    int     py, w;
122:    double  ymin, ymax;
123:    char    tmp[16];
124:
125:    getMinMaxDouble( y, n, &ymin, &ymax );
126:    py = HEIGHT - BOTTOMMARGIN - HEIGHT1 * (y1 - ymin) / (ymax - ymin);
127:    line( LEFTMARGIN, py, WIDTH-1, py, p, 0x5555 );
128:    sprintf( tmp, "%.1f", y1 );
129:    w = strlen( tmp );
130:    symbol( LEFTMARGIN-2-w*6, py-6, tmp, 1, 1, 0, p, 0 );
131:    return;
132: }
133:
134: void       plotDataIntInt( x, y, n, p )
135: int        x[], y[], n, p;
136: {
137:    int     xmin, xmax, ymin, ymax, i, px1, py1, px2, py2;
138:
139:    getMinMaxInt( x, n, &xmin, &xmax );
140:    getMinMaxInt( y, n, &ymin, &ymax );
141:    for ( i = 0; i < n; i++ ) {
142:            px2 = LEFTMARGIN + WIDTH1 * (x[i] - xmin) / (xmax - xmin);
143:            py2 = HEIGHT - BOTTOMMARGIN - HEIGHT1 * (y[i] - ymin) / (ymax -
ymin);
144:            fill( px2-2, py2-2, px2+2, py2+2, p );
145:            if ( i != 0 ) line( px1, py1, px2, py2, p, 0xFFFF );
146:            px1 = px2; py1 = py2;
147:    }
148:    return;
149: }
150:
151: void       plotDataIntDouble( x, y, n, p )
152: int        x[], n, p;
153: double     y[];
154: {
155:    int     xmin, xmax, i, px1, py1, px2, py2;
156:    double  ymin, ymax;
157:
158:    getMinMaxInt( x, n, &xmin, &xmax );
159:    getMinMaxDouble( y, n, &ymin, &ymax );
160:    for ( i = 0; i < n; i++ ) {
161:            px2 = LEFTMARGIN + WIDTH1 * (x[i] - xmin) / (xmax - xmin);
162:            py2 = HEIGHT - BOTTOMMARGIN - HEIGHT1 * (y[i] - ymin) / (ymax -
ymin);
163:            fill( px2-2, py2-2, px2+2, py2+2, p );
164:            if ( i != 0 ) line( px1, py1, px2, py2, p, 0xFFFF );
165:            px1 = px2; py1 = py2;
166:    }
167:    return;
168: }
169:
170: void       plotDataDoubleInt( x, y, n, p )
171: int        y[], n, p;
172: double     x[];
173: {
174:    int     ymin, ymax, i, px1, py1, px2, py2;
175:    double  xmin, xmax;
176:
```

```
177:    getMinMaxDouble( x, n, &xmin, &xmax );
178:    getMinMaxInt( y, n, &ymin, &ymax );
179:    for ( i = 0; i < n; i++ ) {
180:            px2 = LEFTMARGIN + WIDTH1 * (x[i] - xmin) / (xmax - xmin);
181:            py2 = HEIGHT - BOTTOMMARGIN - HEIGHT1 * (y[i] - ymin) / (ymax -
ymin);
182:            fill( px2-2, py2-2, px2+2, py2+2, p );
183:            if ( i != 0 ) line( px1, py1, px2, py2, p, 0xFFFF );
184:            px1 = px2; py1 = py2;
185:    }
186:    return;
187: }
188:
189: void         plotDataDoubleDouble( x, y, n, p )
190: int          n, p;
191: double       x[], y[];
192: {
193:    int     i, px1, py1, px2, py2;
194:    double  xmin, xmax, ymin, ymax;
195:
196:    getMinMaxDouble( x, n, &xmin, &xmax );
197:    getMinMaxDouble( y, n, &ymin, &ymax );
198:    for ( i = 0; i < n; i++ ) {
199:            px2 = LEFTMARGIN + WIDTH1 * (x[i] - xmin) / (xmax - xmin);
200:            py2 = HEIGHT - BOTTOMMARGIN - HEIGHT1 * (y[i] - ymin) / (ymax -
ymin);
201:            fill( px2-2, py2-2, px2+2, py2+2, p );
202:            if ( i != 0 ) line( px1, py1, px2, py2, p, 0xFFFF );
203:            px1 = px2; py1 = py2;
204:    }
205:    return;
206: }
```

リスト5 graphlib.c

```
 1: /*
 2:    graphlib.c
 3:    - baslib のグラフィック関数の一部の代わり
 4:    Jun. 1993        丹 明彦
 5: */
 6:
 7: #include    <iocslib.h>
 8:
 9: void        screen( size, mode, res, disp_sw )
10: int         size, mode, res, disp_sw;
11: {
12:    int     m;
13:
14:    if ( res != 0 && res != 1 ) return;      /* error */
15:    switch ( size ) {
16:    case 0: /* 256x256 */
17:            m = mode * 4 + 2 + 1 - res;
18:            break;
19:    case 1: /* 512x512 */
20:            m = mode * 4     + 1 - res;
21:            break;
22:    case 2: /* 768x512 */
23:            if ( mode != 0 ) return;         /* error */
24:            if ( res == 0 ) return;          /* error */
25:            m = 16;
26:            break;
27:    default:
28:            return; /* error */
29:    }
30:    CRTMOD( m );
31:    if ( disp_sw == 1 ) G_CLR_ON();
32:    return;
33: }
34:
35: void        line( x1, y1, x2, y2, p, ls )
36: int         x1, y1, x2, y2, p, ls;
37: {
38:    static struct   LINEPTR l;
39:    l.x1 = x1;
40:    l.y1 = y1;
41:    l.x2 = x2;
42:    l.y2 = y2;
43:    l.color = p;
44:    l.linestyle = ls;
45:    LINE( &l );
46:    return;
47: }
48:
49: void        fill( x1, y1, x2, y2, p )
50: int         x1, y1, x2, y2, p;
51: {
52:    static struct   FILLPTR f;
53:    f.x1 = x1;
54:    f.y1 = y1;
55:    f.x2 = x2;
56:    f.y2 = y2;
57:    f.color = p;
58:    FILL( &f );
59:    return;
60: }
61:
62: void        symbol( x, y, st, h, v, mo, p, an )
63: int         x, y, mo, p, an;
64: char        *st, h, v;
65: {
66:    static struct   SYMBOLPTR       s;
67:    s.x1 = x;
68:    s.y1 = y;
69:    s.string_address = st;
70:    s.mag_x = h;
71:    s.mag_y = v;
72:    s.color = p;
73:    s.angle = an;
74:    s.font_type = mo;
75:    SYMBOL( &s );
76:    return;
77: }
```

リスト6 gasstat.h

```
 1: /*
 2:  * gasstat.h
 3:  * - 燃費の統計処理みたいなもの
 4:  * Jun. 1993        丹 明彦
 5:  */
 6:
 7: #ifndef     GASSTAT_H
 8: #define     GASSTAT_H
 9:
10: extern int nRecharge;
11:
12: /* ファイルに記述する量 */
13: extern int year[];                          /* 年 */
14: extern int month[];                         /* 月 */
15: extern int date[];                          /* 日 */
16: extern double       odoMeter[];                     /* 総走行距離 */
17: extern double       gasoline[];                     /* 給油量 */
18: extern int money[];                         /* 価格 */
19:
20: /* ファイルに記述しない(あとから算出する)量 */
21: extern double       tripMeter[];                    /* 走行距離 */
22: extern double       netDistance[];                  /* 純総走行距離 */
23: extern double       totalGasoline[];                /* 総給油量 */
24: extern double       netTotalGasoline[];             /* 純総給油量 */
25: extern double       nenpi[];                        /* 燃費 */
26: extern double       averageNenpi[];                 /* 平均燃費 */
27: extern double       price[];                        /* 1lあたりの値段 */
28: extern int day[];                           /* 使用日数 */
29:
30: void        readData( FILE * );
31: void        calcData();
32: void        printData();
33:
34: #endif      /* GASSTAT_H */
```

リスト7 graphlib.h

```
 1: /*
 2:    graphlib.h
 3:    - baslib のグラフィック関数の一部の代わり
 4:    Jun. 1993        丹 明彦
 5: */
 6:
 7: #ifndef     GRAPHLIB_H
 8: #define     GRAPHLIB_H
 9:
10: void        screen( int, int, int, int );
11: void        line( int, int, int, int, int, int );
12: void        fill( int, int, int, int, int );
13: void        symbol( int, int, char *, char, char, int, int, int );
14:
15: #endif      /* GRAPHLIB_H */
```

リスト8 plot.h

```
 1: /*
 2:     plot.c
 3:     - グラフ描画
 4:     Jun. 1993        丹 明彦
 5: */
 6:
 7: #ifndef     PLOT_H
 8: #define     PLOT_H
 9:
10: void        plotAxis( int );
11: void        getMinMaxInt( int*, int, int*, int* );
12: void        getMinMaxDouble( double*, int, double*, double* );
13: void        plotNotchXInt( int*, int, int, int );
14: void        plotNotchXDouble( double*, int, double, int );
15: void        plotNotchYInt( int*, int, int, int );
16: void        plotNotchYDouble( double*, int, double, int );
17: void        plotDataIntInt( int*, int*, int, int );
18: void        plotDataIntDouble( int*, double*, int, int );
19: void        plotDataDoubleInt( double*, int*, int, int );
20: void        plotDataDoubleDouble( double*, double*, int, int );
21:
22: #endif      /* PLOT_H */
```

リスト9 compile.bat

```
echo off

rem COMPILE.BAT
rem 最低限の処理，エラーチェックもない
rem できるだけ make を使うこと

set lib=a:¥lib
set include=a:¥include
cc /O /Fc nenpi_t.c
cc /O /Fc nenpi_g.c
cc /O /Fc gasstat.c
cc /O /Fc plot.c
cc /O /Fc graphlib.c
cc nenpi_t.o gasstat.o
cc /Y nenpi_g.o gasstat.o plot.o graphlib.o
```

開発効率向上のため

makeを使おう

Tan Akihiko 丹　明彦

以前からmakeがいかに便利なツールかを訴える人はいましたが，Xc ver.1にはついていなかったためか，コンパイルをバッチファイルなどで済ます人もまだ多いようです。この機に一気に文化的開発環境を整えましょう。

すべてのCプログラマはmakeを使わねばならぬ。makeは単にコンパイル手順を記述するバッチファイルもどきではない。もっと賢くて器用で気が利いている。要するにmakeを使えば世の中バラ色ってやつなのだ。『Life with UNIX』にも語られているとおり，makeを使わないと天罰が下ると思っておいたほうがいいだろう。

makeとはなにか

makeはXCプログラマーズマニュアルによれば「ファイル保守ユーティリティ」と紹介されており，それはそれでまったく正しいのだが，あまりおいしそうなツールに見えないというのはいただけない話だ。リファレンスマニュアルゆえしかたのないことではあるが，説明を読んでもいまいちピンとこない。

makeは巨大で依存関係の複雑なソースファイルを間違いなくコンパイルするためのツールである。ソフトウェアの開発者にとっては，余計なことを考えずにプログラミングに集中させてくれるための手ばなせないツールである。フリーソフトウェアをもらってきて利用する人にとっては，作者と共通のコンパイル環境を一切の面倒なく得ることができるツールでもある。

makeはどこで手に入るのか

たとえばパソコン通信でしか入手できないツールがあって，これ便利だよと人にさんざん宣伝しておいて，いざどうしたら手に入るのと聞かれたときに,「どこかにあるはずですから頑張って手に入れてくださいね」とだけ答えたら顰蹙を買ってもしかたない。ソフトウェア配布というのはなかなかに難しい問題だ。通信回線を通じてしか配布していないものを「available」といっていいのは事実上インタネット(主にUNIXユーザーで構成されている全世界規模のネットワークで，そのパフォーマンスはパソコン通信の比ではない。それを利用できるのは大学などの研究機関がほとんどで，個人ユーザーで利用しているという例はまずないと思ってよい）だけではあるまいか。

話がそれたが，幸いにして，makeを手に入れるのは容易である。

XCバージョン2以降に標準添付
ソフトバンク刊『X68k Programming Series #2 X680x0 libc』(通称libc)に添付

つまり，Cコンパイラを使える環境にある人のほとんどは，makeをすでに持っているのである。

makeはどのような場面で有効か

燃費計算プログラムを例に取ろう。

図1はソースファイルがどのようにコンパイルされてオブジェクトファイルおよび実行ファイルに変化していくかを示した図である。簡単にいえば，5つのCで書かれたソースファイルがあり，それぞれがコンパイルされてオブジェクトファイルになり，結合されることによって実行ファイルになる。これはいってみれば，実行ファイルを無事に生成するための文法的な依存関係を示している。

これとは別に，ソースファイルの間には，意味的な依存関係もある。テキストしか使わないnenpi_t.cは燃費計算のキモであるgasstat.cの関数しか呼んでいないため，インクルードするファイルは(標準ヘッダファイルであるstdio.hを除けば)gasstat.cを利用するための設定を収めたgasstat.hだけである。

しかし，グラフィックを使うほうのnenpi_g.cは，gasstat.cのほかにもグラフィックライブラリであるgraphlib.cやグラフ描画を行うplot.cの関数も呼んでおり，インクルードするファイルもそれに応じてgraphlib.hおよびplot.hが増えている。そ

図1　ファイル生成

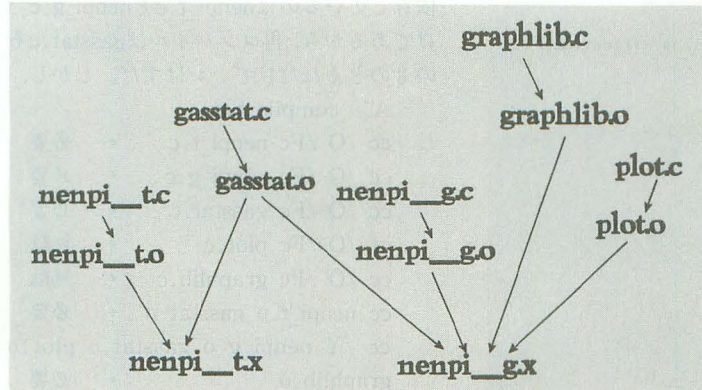
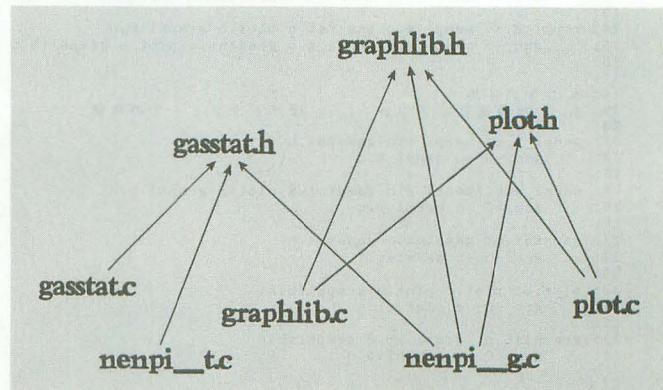



図2　ソースファイルのインクルード

れが図2である。

これの意味するところは，「もしもgasstat.cで使う変数や関数の仕様を変えれば，それに依存するnenpi_t.cやnenpi_g.cを再コンパイルしなくてはならない」ということである。逆に，graphlib.cはgasstat.cには依存していないので，gasstat.cの仕様がどれほど変わろうと，graphlib.cを再コンパイルする必要がまったくないということである。

makeは，こうした依存関係を把握し，プログラマが修正したソースファイルとそれに依存するファイルだけを再コンパイルするために用いるのである。常に必要最小限のコンパイルだけを行うということである。

メイクファイルは，コンパイルの手順だけでなく，こうした依存関係をも記述するファイルである。すなわち，作者のコンパイル環境に関する情報を詰め込んだファイルといってもよい。

makeはどのように使うのか

makeは，デフォルトではMakefile(純正のHuman68kでは大文字と小文字を区別しないのでMAKEFILEでもMaKeFIlEでもいいのだが）という名前のメイクファイルをカレントディレクトリから探す。すなわち，ただコンパイルするだけならば，カレントディレクトリにMakefileという名前のファイルがあることを確認したうえで，

A> make

とやればいいのである。文字どおり「make一発」というやつである。

デフォルトでない名前のファイルをメイクファイルとして与えたい場合は，-fオプションをmakeに与える。これは，

A> make -f forX68030.mak

のようにして，X68030用に書かれたメイクファイルを指定するような場面などに有効である。

バッチファイルでもいいんじゃない

いいえ。

……といっておくだけでもいいのだが，実例を示そう。

私の燃費計算プログラムのリストのページには，コンパイル用にバッチファイルを載せている。もちろん私は開発中にバッチファイルは使っていないのだが，ここに悪意を持ってバッチファイルの欠点を攻撃するものである。

初回のコンパイルは問題ない。

```
A> compile.bat
cc /O /Fc nenpi_t.c
cc /O /Fc nenpi_g.c
cc /O /Fc gasstat.c
cc /O /Fc plot.c
cc /O /Fc graphlib.c
cc nenpi_t.o gasstat.o
cc /Y nenpi_g.o gasstat.o plot.o
graphlib.o
```

問題は，ここで，たとえばgasstat.cの関数の仕様をちょっと変えたような場合だ。関数の仕様はgasstat.hに宣言されるため，当然gasstat.hの書き換えを伴う。

先ほどの依存関係によれば，gasstat.cに依存しているのはnenpi_t.cとnenpi_g.cだけであるから，再コンパイルはgasstat.cそのものとそれだけでいいはずだ。しかし，

```
A> compile.bat
cc /O /Fc nenpi_t.c              ← 必要
cc /O /Fc nenpi_g.c              ← 必要
cc /O /Fc gasstat.c              ← 必要
cc /O /Fc plot.c                 ← 無駄
cc /O /Fc graphlib.c             ← 無駄
cc nenpi_t.o gasstat.o           ← 必要
cc /Y nenpi_g.o gasstat.o plot.o
graphlib.o                       ← 必要
```

リスト1

```
 1: # 基本的なメイクファイル
 2: # XC バージョン2 用, 小技なし
 3:
 4:
 5: # 最終ターゲット
 6: # make のデフォルトのターゲットは all
 7:
 8: all: nenpi_t.x nenpi_g.x
 9:
10:
11: # リンク
12: # ファイルの依存関係を記述した行の次に具体的なコマンドラインを書く
13: # /Y は DOS/IOCS ライブラリを使うためのオプション
14:
15: nenpi_t.x: nenpi_t.o gasstat.o
16:     cc nenpi_t.o gasstat.o
17:
18: nenpi_g.x: nenpi_g.o gasstat.o plot.o graphlib.o
19:     cc /Y nenpi_g.o gasstat.o plot.o graphlib.o
20:
21:
22: # コンパイル
23: # /O は最適化オプション, /Fc はリンクフェイズの抑制
24:
25: nenpi_t.o: nenpi_t.c gasstat.h
26:     cc /O /Fc nenpi_t.c
27:
28: nenpi_g.o: nenpi_g.c gasstat.h plot.h graphlib.h
29:     cc /O /Fc nenpi_g.c
30:
31: gasstat.o: gasstat.c gasstat.h
32:     cc /O /Fc gasstat.c
33:
34: plot.o: plot.c plot.h graphlib.h
35:     cc /O /Fc plot.c
36:
37: graphlib.o: graphlib.c graphlib.h
38:     cc /O /Fc graphlib.c
```

リスト2

```
 1: # 基本的なメイクファイル
 2: # gcc 用, 小技なし
 3:
 4: all: nenpi_t.x nenpi_g.x
 5:
 6:
 7: # リンク
 8: # -o ファイル名 は出力ファイル名(実行ファイル名)の指定
 9: # -lfloatfnc は XC ライブラリのためのもの (libc では必要ない)
10:
11: nenpi_t.x: nenpi_t.o gasstat.o
12:     gcc -o nenpi_t.x nenpi_t.o gasstat.o -lfloatfnc
13:
14: nenpi_g.x: nenpi_g.o gasstat.o plot.o graphlib.o
15:     gcc -o nenpi_g.x nenpi_g.o gasstat.o plot.o graphlib.o -liocs -lfloatfnc
16:
17:
18: # コンパイル
19: # -O は最適化オプション, -c はリンクフェイズの抑制
20:
21: nenpi_t.o: nenpi_t.c gasstat.h
22:     gcc -O -c nenpi_t.c
23:
24: nenpi_g.o: nenpi_g.c gasstat.h plot.h graphlib.h
25:     gcc -O -c nenpi_g.c
26:
27: gasstat.o: gasstat.c gasstat.h
28:     gcc -O -c gasstat.c
29:
30: plot.o: plot.c plot.h graphlib.h
31:     gcc -O -c plot.c
32:
33: graphlib.o: graphlib.c graphlib.h
34:     gcc -O -c graphlib.c
```

ということで，2つの無駄なコンパイルを行っている。これは，バッチファイルがファイルの依存関係をまったく感知しないために起こったことである。

これが，一般的なフリーソフトウェアにおいては，数10個から数100個にも及ぶ，複雑に絡みあった依存関係を持つソースファイルの集まりになるのである。これをきっちり管理するというのは大変である。まして作者でもない人間が依存関係をきっちり調べ上げるのはほとんど不可能なことであろう。

この，作者しか知らない知識を簡潔かつ強力に管理できるのがmakeのいいところなのである。

メイクファイルの例

ここに4つのメイクファイルを挙げた。（リスト1～4）それぞれ，

正直に書いたXC用メイクファイル
正直に書いたgcc用メイクファイル
小技を使ったXC用メイクファイル
小技を使ったgcc用メイクファイル

という構成になっている。このうち1番目と3番目について軽く解説を試みる。コメントもつけてあるのであわせて参照されたい。

●リスト1（正直に書いたXC用メイクファイル）

メイクファイルの基本は，

ターゲット: 依存ファイル
　　　　コマンドライン

である。依存ファイルは複数あってもかまわないし，コマンドラインは省略することもできる。あと，これはmakeのルールだが，コマンドラインはタブで始める必要がある。

このメイクファイルはいたって素直な作りになっており，最終的に作るのは2つの実行ファイルで，それを作るためにはどれとどれをコンパイルして，といったように順番に見ればすぐわかる。依存関係を記述した行に注目してほしい。

●リスト3（小技を使ったXC用メイクファイル）

リスト1を見ると，コマンドラインの多くが同じ始まり方をしている。コンパイラの起動のしかたが共通ということだ。これをいちいち書くのは面倒である，ということで，リスト3の途中を見てほしいのだが，%とか$とかいった，ふだんCコンパイラを扱っているときには見かけない妙な記号がある。ちょっとした技である。

このおかげで，リスト2の末尾の依存関係を記述する部分が依存関係のみを記述するようになっている（コマンドラインがない）。これでもちゃんと，更新したファイルに依存しているファイルが%などにパターンマッチするため，再コンパイルされるのである。

あとはおまけで，cleanなどのエントリを入れてみた。これはフリーソフトウェアで配布されているもののなかにときどき見かけるもので，コンパイルが成功したあとにmake cleanとやると，途中で生成した実行時には不要なオブジェクトファイルだけを消してくれる。さらに，make cleandistとやると，生成した実行ファイルも消してくれ，配布時の状態に戻る。

おわりに

makeは汎用の開発支援ツールである。Cに限らず，アセンブラだろうがPascalだろうが，TeXの処理だって書けてしまう。

makeはプログラムを配付するときには特に便利だが，個人的なプログラムでもMakefileは書いておいたほうがよい。昔作ったプログラムをコンパイルしようとするときに重宝するからだ。1年前に作ったプログラムの，ソースファイルの依存関係を覚えているという人間はそうはいない。X年後の自分は他人，という格言（？）が実感としてわかる人は，メイクファイルを書く習慣を身につけることをおすすめする。

リスト3

```
 1: # 小技の入ったメイクファイル
 2: # XC バージョン2, シャープ純正環境
 3:
 4: all: nenpi_t.x nenpi_g.x
 5:
 6: nenpi_t.x: nenpi_t.o gasstat.o
 7:     cc nenpi_t.o gasstat.o
 8:
 9: nenpi_g.x: nenpi_g.o gasstat.o plot.o graphlib.o
10:     cc /Y nenpi_g.o gasstat.o plot.o graphlib.o
11:
12:
13: # コンパイル規則
14: # % には下のファイル依存関係でマッチしたファイル名が入る
15: # $< は依存ファイルの最初のファイル名が入る
16: # オブジェクトファイル (*.o) を C のソースから生成する規則
17:
18: %.o: %.c
19:     cc /O /Fc $<
20:
21:
22: # お掃除
23: # make clean でテンポラリファイルを消す
24: # make cleandist ですべての生成ファイルを消す
25:
26: clean:
27:     if exist nenpi_t.o del nenpi_t.o
28:     if exist nenpi_g.o del nenpi_g.o
29:     if exist gasstat.o del gasstat.o
30:     if exist plot.o del plot.o
31:     if exist graphlib.o del graphlib.o
32:
33: cleandist: clean
34:     if exist nenpi_t.x del nenpi_t.x
35:     if exist nenpi_g.x del nenpi_g.x
36:
37:
38: # ファイルの依存関係
39:
40: nenpi_t.o: nenpi_t.c gasstat.h
41: nenpi_g.o: nenpi_g.c gasstat.h plot.h graphlib.h
42: gasstat.o: gasstat.c gasstat.h
43: plot.o: plot.c graphlib.h
44: graphlib.o: graphlib.c graphlib.h
```

リスト4

```
 1: # 小技の入ったメイクファイル
 2: # UNIX風ツール (ここでは rm) が必要
 3:
 4: all: nenpi_t.x nenpi_g.x
 5:
 6: nenpi_t.x: nenpi_t.o gasstat.o
 7:     gcc -o nenpi_t.x nenpi_t.o gasstat.o -lfloatfnc
 8:
 9: nenpi_g.x: nenpi_g.o gasstat.o plot.o graphlib.o
10:     gcc -o nenpi_g.x nenpi_g.o gasstat.o plot.o graphlib.o -liocs -lfloatfnc
11:
12: %.o: %.c
13:     gcc -O -c $<
14:
15: clean:
16:     rm -f nenpi_t.o nenpi_g.o gasstat.o plot.o graphlib.o
17:
18: cleandist: clean
19:     rm -f nenpi_t.x nenpi_g.x
20:
21: nenpi_t.o: nenpi_t.c gasstat.h
22: nenpi_g.o: nenpi_g.c gasstat.h plot.h graphlib.h
23: gasstat.o: gasstat.c gasstat.h
24: plot.o: plot.c graphlib.h
25: graphlib.o: graphlib.c graphlib.h
```

コマンドシェル制作過程

プログラムの書き方

Nakamori Akira 中森 章

そして最後はプログラミングの方法論です。どのようなアプローチでプログラムを作っていくのか，どんな手法があるのか，実際のプログラミング現場からの実況でご覧ください。

はじめに

相変わらずのJリーグ人気です。「読売」を冠して呼ばれることの多いヴェルディよりも，企業色の薄いエスパルスが私のお気にいりです。『ちびまる子ちゃん』に出てきた健気なケン太くんがサブリミナル効果として作用してるのかもしれませんが。ところで，日本テレビで放映されるヴェルディの試合は，アナウンサーがやたら「ウラァモスゥ！」とか「カズゥー！」を連呼するので聞き苦しいと思いませんか？

さて，C言語に対するニーズの高さを反映してか，巷にはC言語の参考書があふれています。コンピュータに関係する仕事をしている人ならば，1冊はC言語の参考書を持っているようです。

しかし，その割にはC言語でプログラムを書いている人を見ることは希です。このギャップはどこから生まれるのでしょう。おそらく，プログラムを書きたいと思っても，具体的にどうしたらよいかわからないというのが実情でしょう。

ほとんどすべての参考書（特に入門書）が，C言語の文法とプログラム例とその解説で構成されています。その反面，そのプログラム例がどのようにして導かれたのかについて説明されているものは少数です。丁寧な解説によって，プログラム例がC言語の文法に沿っていること（当たり前）や与えられた機能を満足するものであることは理解できるでしょうが，多くの場合はそれだけで終わりです。そのプログラム例のエッセンスを自分のプログラミングに応用することのできる人は高度なプログラムセンスの持ち主だと自負してよいでしょう。

C言語に限らずプログラミングには一種のセンスが必要です。それは，新たなテーマに直面したときに，過去の経験を参照したり，新規に有用な情報を探し当てて，ひとつのプログラムとしてまとめあげる能力です。このセンスを磨くことこそが本当のプログラミング言語の理解というものではないでしょうか（その点ほとんどの入門書は失格だな）。

これはプログラミングの場数を踏むことで身につけていくものです。とはいえ，具体的な手法が示されなくては場数を踏むことさえままならないでしょう。ということで，今回は，あるテーマが与えられてからプログラムとして完成するまでにどのような思考の流れがあるのか，私自身の例を示しながらケーススタディしていきましょう。もちろんこれがプログラムを書く手法のすべてとはいいません。しかし，どのようにプログラムを書いたらよいのかわからない人には，多少なりとも参考になるでしょう。

なにを作るか

プログラムを書くということは，なにかコンピュータにさせたい処理があるということです。多くの場合，こんなことができたらいいな，という発想の下にプログラムの作成が開始されます。今回はちょっと難しそうなものをということで，コマンドシェルを作ってみたいと思います。

ところで，シェルとひと口にいってもいろいろな種類があります。ここでイメージしているのはCOMMAND.Xに毛が生えた程度（実はサブセットかもしれない）のコマンド起動プログラムです。その機能として，

　入力したコマンドが起動できる
　最低限の組み込みコマンドがある
　コマンドラインの編集ができる
　簡単なヒストリ機構がある

を最低限の目標にしたいと思います。しかし，これだけでは実用性に欠けるので，

　ファイル名の補完機能
　コマンド名の補完機能
　標準入出力のリダイレクト機能

を持たせましょう。

補完というのはcompletion（完全化）を訳したもので，ファイル名の先頭の数文字を与えると，それに一致するファイルの集合を表示し，もし一致するファイルがひとつならそのファイル名をコマンドラインに埋め込むという機能です。これは，UNIXの各種コマンドシェルでは大変にポピュラーな機能です。COMMAND.Xでもver.3.0から採用されました（TABや^Xで使用できる）。

また，標準入出力のリダイレクトは，C言語で書かれたプログラムを実行するためには必須の機能です。標準入出力のリダイレクトができないシェルなんてシェルじゃありません。

ところで，シェルというからにはシェルスクリプト（早い話がバッチファイル）用の制御構造がなければ不十分です。しかし，今回はプログラムが巨大化しそうなので実現はやめておきます。同じ理由でパイプ機能やマルチステートメントも見送ります。

実現したい機能をはっきりさせよ

おおまかな構成を考える

作るテーマが決まったら，次はプログラムのおおまかな構成を考えなければなりません。これはプログラムをいくつかの機能ブロックに切り分ける作業です。たとえば上述のコマンドシェルなら，

　1行入力する
　入力されたコマンドを実行する

ということが最小の機能ブロックになります。これを実現する機能に従って細分化していけばよいのです。

ヒストリ機能やコマンドラインの編集は1行入力の範疇です。ファイル名やコマンド名の補完も1行入力に付随する機能です。

一方，組み込みコマンドの実行は1行入力されたあとの処理です。また，入力された1行のヒストリへの記録や標準入出力のリダイレクトは，1行入力とコマンド実行の中間にある処理です。このようなことを考えると，今回のシェルのおおまかな構成は，

　1行入力　　行の編集
　　　　　　　ヒストリから呼び出し
　　　　　　　ファイル名の補完
　　　　　　　コマンド名の補完
　最新行をヒストリへ記録
　リダイレクト処理を行う
　コマンド実行　組み込みコマンド処理
　　　　　　　　その他のコマンド処理

の繰り返し処理ということになります。

ここでは具体的なC言語での記述を意識する必要はありません。行の編集とかファイル名の補完とかいっても，まだまだ雲をつかむような処理だと思います。しかし，コマンドシェルという途方もないテーマから比べると少し具体性が出てきたような気がしませんか。

さて，プログラムをいくつかの機能ブロックに分けることができたら，さらにそれぞれをひとつのプログラムと考えてさらに処理の細分化を行うことを考えます。この段階ではC言語でどのように実現するかということを少しは考慮していなければなりません。

機能ブロックをどんどん細分化して考えていくと，もうこれ以上分けられない処理，あるいは実現方法が明らかでそれ以上考えなくてよい処理に突き当たります。それが，その機能ブロックでのキーポイントとなる処理です。プログラミングとはこのキーポイントの処理を中心に肉づけを行う作業なのです。すなわち，

1)　まずキーポイントとなるプログラムを書いてprog0とする。

2)　prog0を呼び出す（利用する）プログラムを書いてprog1とする。

⋮

3)　prog(n-1)を呼び出す（利用する）プログラムを書いてprog(n)とする。

4)　prog(n)を呼び出す（利用する）プログラムを書けば機能ブロックを実現するプログラムができあがる。

5)　機能ブロックを実現するプログラムを寄せ集めれば目的のプログラムになる。

という具合です。

これは，従来，金科玉条のようにいわれてきた構造化プログラミングとは逆の立場のボトムアップなやり方です。構造化プログラミングでは，トップダウンにサブルーチンを決定していき，最下層のサブルーチンが作成されたときにはプログラムが完成しています。しかし，私が実践している方法は最下層から組み上げていく手法です。

プログラミングの詳細仕様が決定したあとにプログラミング（コーディング）をするのならどちらでも大差ありません。しかし，多分に行き当たりばったり的な性格を帯びた（私の）プログラミングでは，上位のサブルーチンとのインタフェイスをあとから決定できるため，ボトムアップな手法のほうが便利なのです。

トップダウンな手法では上位のほうからインタフェイスがすでに規定されているので，設計を誤ると最初からやり直しということになりかねません。

どんな大きなプログラムも小さいプログラムの組み合わせ

機能ブロックの実現方法を考える

機能ブロックの実現はキーポイントとなるプログラムを書けるか否かにかかっています。ここではキーポイントとなるプログラムをどのようにして作成するのか，以下に例を示しましょう。これには，少なくとも3つの方法があります。

1)　長年の経験にものをいわせて力でねじ伏せる。

2)　似たような機能を有するものを，論文，雑誌，あるいは他人のプログラムから見つけてきて，目的にあわせて改造する。

3)　マニュアルの中から最適なライブラリを探して組み合わせる。

2)や3)は似たようなプログラムをどこかから探してくること，1)は自分や他人が書いたプログラムの中で似たような処理をしていたのを思い出すことです。そして，これらの繰り返しが場数を踏むということです。プログラミングが上手な人とは，このキーポイントを見つけるのが早く，さっさとC言語で書き下ろしてしまえる人だと思います。

ところで，現実のプログラミングでは，キーポイントの部分の処理がこれまでに見たことのないものの場合もあります。この場合は2)や3)の方法を採るしかしかたありません。こんなとき，目的にあいそうなライブラリをマニュアルの中に見つけてもその動作がよくわからないことがあります（特に使用例が載っていない場合は）。

しかし，ここで，わからないよと投げ出してはいけません。小さなプログラムを書いてライブラリの動作を理解するようにしなければなりません。さすがに，このような状態で書いたプログラムでのバスエラーは暴走を引き起こす確率が高いのですが，これがのちのちのための経験の蓄積になるのです。

もっともキーポイントとなる部分から考えろ。あとは肉づけだけ

新しい試みのときは小さなプログラムで実験して動作を理解せよ

参考書のプログラム例を見るときは，将来自分で使用することを想定してみること

シェルの基本機能

プログラミングの方針がはっきりしたところで，いよいよ実際にプログラムを書いてみましょう。

シェルといえば入力した行をコマンドとして実行するプログラムであることはすでに示しました。ここでのキーポイントは，

　1行入力する部分
　プログラムを実行する部分

であることは明らかです。そのプログラムの実現はC言語のライブラリを使用すれば簡単で，

　1行入力　→　gets
　実行する　→　system

でよいでしょう。

system関数は与えられた文字列をCOMMAND.Xのコマンドラインとして実行します。このとき，プログラムはリスト1のようになります。短いプログラムですからなにをしているのかは明白ですね。コンパイルしていろいろな入力を与えて実行してみてください。プログラムを終了するためにはEOF（CTRL＋Z）を入力します。これはsystem関数の存在を知っていればすぐに完成するプログラムですね。

しかし，リスト1のプログラムには不満があります。それは，COMMAND.Xの組み込みコマンドを実行しない場合でもCOMMAND.Xが起動されるという点です。そこで，直接コマンドを実行できるライブラリ関数はないかと探します。「実行」を英語でいうと「execution」ですから，execに当た

りをつけてマニュアルを探すと，

execl, execle, execlp, ...

などという一連の関数を見つけることができます。

説明によるとコプロセスを起動して制御を移す関数ということになっています。それならば，これらがsystem関数の代わりに使えるかもしれません。しかし，サンプルプログラムを作って実験してみればわかりますが，これらの関数を呼び出したあとは呼び出し側に制御が戻りません。1回コマンドを実行したらおしまいというわけです。これではシェルに使用することはできませんね。

リスト1

```
 1: /*
 2:     コマンドシェルの基本
 3: */
 4: #include <stdio.h>
 5:
 6: char LINE[1024];
 7:
 8: int get_line(void)
 9: {
10:     printf("$ ");
11:     return (int)gets(LINE);
12: }
13:
14: main()
15: {
16:     while(get_line()!=NULL){
17:         if(LINE[0]==0  ) continue;
18:         system(LINE);
19:     }
20: }
21:
```

そこで目につくのがexecl関数などの参照関数として載っているspawnナンチャラという関数です。さっそく，マニュアルでそのページを見ます。説明は子プロセスを新たに作成して実行する関数ということです。マニュアルには明記されていませんが，こちらは呼び出し側に制御が戻ってくるようです。この関数を使ってリスト1を書き換えることにしましょう。spawnナンチャラはC言語のmain関数に，

main(int argc,char ＊argv []) {

のように渡されてくるargvという引数の形式の引数を作って呼び出せばよいようです。ここでは引数の個数のいちばん少ないspawnvを使用することにしましょう。

さて，spawnvを使用するためにはargvに相当する引数を作り出してやる必要があります。これは入力されてきたコマンドラインを，その文字列に含まれる単語（部分文字列）に分解して，それぞれの単語が格納されている領域のアドレスを要素とする配列を作ることです。これは，イメージ的には図1に示すような処理になります。しかし，ここでは単語を格納する領域をわざわざ確保することはせず，コマンドラインを直接その領域に割り当てるということをしてみました。

コマンドラインの単語と単語の間には必ず1個以上の空白文字があるはずなので，そこにヌル文字を書き込んで部分文字列を作っているのです（図2）。

単語の切り分けを行い，コマンド実行の処理にspawnv関数を使用してリスト1を書き換えるとプログラムはリスト2のようになります。これにより，無駄にCOMMAND.Xが呼び出されることはなくなりましたが，失った機能もあります。今度はdirやcopyなどのCOMMAND.Xの組み込みコマンドが使えなくなってしまいました。まさか，

COMMAND.X　DIR

のように，いったんCOMMAND.Xを起動することを強要することもできませんから，COMMAND.Xの組み込みコマンドは今回のシェルの組み込みコマンドとして実現してやるしかなさそうです。

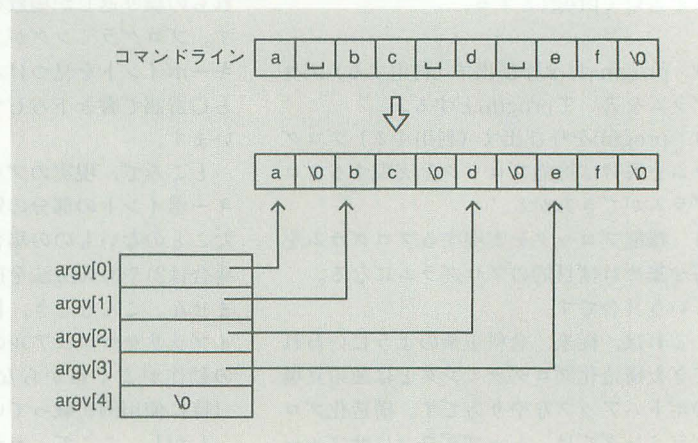
組み込みコマンドを実装してみる

組み込みコマンドの実行時，ユーザーにCOMMAND.Xを起動することを強要はできません。しかし，その操作をシェルが勝手にやるとしたら話は別です。dirやcopyと

リスト2

```
 1: /*
 2:     コマンドシェルの基本
 3: */
 4: #include <stdio.h>
 5: #include <process.h>
 6:
 7: char LINE[1024];
 8: int  argc;
 9: char *argv[100];
10:
11: /* 入力されたコマンドラインを単語に切り分ける処理 */
12: void split(void)
13: {
14:     char *bgn,*end;
15:
16:     bgn = LINE;
17:     argc = 0;
18:     while(1){
19:         while((*bgn==' ')||(*bgn=='¥t')) bgn++;
20:         end = bgn;
21:         if((*bgn==0)||(*bgn=='¥n')) break;
22:         while((*end!=' ' )&&(*end!='¥t')
23:             &&(*end!='¥n')&&(*end!=0    )) end++;
24:         argv[argc++] = bgn;
25:         if(*end==0) break;
26:         *end = 0;
27:         bgn = ++end;
28:     }
29:     *end = 0;
30:     argv[argc] = 0;
31: }
32:
33: int get_line(void)
34: {
35:     printf("$ ");
36:     return (int)gets(LINE);
37: }
38:
39: main()
40: {
41:     while(get_line()!=NULL){
42:         if(LINE[0]==0  ) continue;
43:         split();
44:         spawnv(P_WAIT,argv[0],(const char**)argv);
45:     }
46: }
47:
```

図1　コマンドラインを切り分ける(その1)

図2　コマンドラインを切り分ける(その2)

いった組み込みコマンドが入力されたら，コマンドラインを，

COMMAND.X　DIR

COMMAND.X　COPY　file1 file2

などと読み換えてspawnv関数に引き渡すことで，組み込みコマンドを実現することにしましょう。この処理はargvの内容をひとつずらして，argv［0］に”COMMAND.X”という文字列のアドレスを与えてspawnvを呼び出すことになります。

さて，コマンドラインから入力されてくるコマンドが組み込みコマンドかそれ以外かを調べるにはどうしましょう。コマンドラインが入力され，単語への切り分けが終わったあと，strcmp関数でargv［0］が組み込みコマンドであるか否かを比較するだけですね。コマンド名の大文字小文字を区別したくないならstrcmpi関数で比較するだけです。具体的には，

if(strcmp(argv［0］,”dir”)==0)｛……｝

if(strcmp(argv［0］,”copy”)==0)｛……｝

if(strcmp(argv［0］,”del”)==0)｛……｝

のようにif文をひたすら並べまくるだけです。strcmp関数の戻り値が0ならば組み込みコマンドとして扱うのです。

上のように処理すればCOMMAND.Xの組み込みコマンドをなんでも実行できるような気がします。しかし，少しは苦労するところもほしいのでcdコマンドくらいはシェルの側で制御することにしましょう。cdは明らかにCOMMAND.X（HUMAN.SYS）のコマンドなのでマニュアルのDOSライブラリを探します。ディレクトリ関係のライブラリではドライブを変更するための，

CHGDRV

ディレクトリを変更するための，

CHDIR

が使えそうです。

COMMAND.Xではドライブを変更する場合はドライブ名（：つき）のみを使用し，cdはディレクトリを変更するためだけに使用します。今回のシェルではドライブ名もディレクトリの一部と考えて，ドライブのみの変更も必ずcdをつけて，

cd　A:

などのように入力することにしましょう。これを実現するためのキーポイントとしてはディレクトリを示す文字列の中でドライブ名とディレクトリを切り分けて，それぞれ独立に変更することです。

CHGDRV関数やCHDIR関数をどのように使い分けるかを試すためのプログラムがリスト3です。リスト3のプログラムをコンパイルするためにはdoslib.aまたはdoslib.lをリンクする必要があります。ディレクトリが正しく変更できたことを確かめるために，リスト3ではディレクトリを変更するごとにsystem関数でdirコマンドを実行しています。

リスト3でいろいろと実験しているとわかりますが，ドライブの変更では存在しないドライブに移行しようとしても，それがCONFIG.SYSで指定したLASTDRIVEに含まれるならエラーになりません。当然，ドライブは変更されません。実際のシェルではこれを回避する手段が必要でしょう。

行の編集をどうするか

ここまではコマンドの実行をメインに考えてきました。そろそろ1行入力について考えましょう。ここで問題になるのはコマンドラインをどのようにして編集するかということです。

それではコマンドラインを編集するためのキーポイントを考えていきましょう。とはいえ，具体的な手法がはっきりしませんね。こういう場合は基本に立ち返りましょう。コマンドラインの編集とは文字列に対して，その一部を別の文字で置き換えたり，ある位置に別の文字を挿入したり，文字を削除したりすることです。

これまでの例を見ればわかるように，コマンドラインを示す文字列を取り込むにはgets関数を使用すればいいことがわかっています。ところが，gets関数では改行コードを入力するまで入力された文字列を参照することはできません。当然，改行コード以前に入力された文字列を編集することなどは不可能です。

このときひらめくのはgets関数の代わりを自作すればいいということです。つまり，キーボードから入力されてくる文字をすべて管理していて，改行コードに出合ったらそれまでに入力されている文字列を実行部に渡してやればよいのです。入力されてくる文字をすべて知っているので，編集用の特殊なコードが入力されたら，それまでに入力された文字列を編集することができるはずです。

ところで，編集においては入力された文字がどこに挿入されるか（コマンドラインは常に挿入モードで動作することを仮定している）を示すためにカーソルを適当な位置に移動しなければなりません。つまり文字列（char型の配列）に対する文字の挿入位置を保持する変数（添え字）の値に応じて画面上に表示されているカーソルを移動しなければなりません。このカーソル移動はどう実現したらよいのでしょう。

C言語のライブラリを探せばその目的のためのライブラリがあります。しかし，文字挿入などを考えるとどうせ自前で処理する部分を作らなければなりませんので今回はすべて自作することにします。それは，次のような興味深いアイデアを思いついたからです。

1)　カーソルを右に移動するためには，現

リスト3

```
 1: #include <stdio.h>
 2: #include <doslib.h>
 3:
 4: char line[255];
 5:
 6: main()
 7: {
 8:     unsigned char directory[100];
 9:     unsigned char *dir;
10:     int  drv;
11:
12:     while(1){
13:         printf("ディレクトリ=>");
14:         if(gets(line)==NULL) break;
15:         sscanf(line,"%s",directory);
16:         dir = directory;
17:         if(directory[1]==':'){ /* ドライブ名がある */
18:             drv = toupper(directory[0]); /* ドライブ名     */
19:             drv = drv-'A';               /* ドライブ番号   */
20:             dir += 2;                    /* ディレクトリ名 */
21:             if(CHGDRV(drv)<drv){
22:                 fprintf(stderr,"%c: ドライブ変更ができません。¥n",drv+'A');
23:                 continue;
24:             }
25:         }
26:         if(*dir==0) /* ドライブ名のみを指定したとき */
27:             continue;
28:         if((*dir!=0)&&(CHDIR(dir)<0))
29:             fprintf(stderr,"%s: ディレクトリ変更ができません。¥n",dir);
30:         system("dir /W");
31:     }
32: }
33:
34:
```

カーソル位置の文字をプリントすればよい。
2) カーソルを左に移動するためには，カーソルをいったん左端に移動して（キャリッジリターンをプリント）現カーソル位置のひとつ左の文字までをプリントすればよい。

こんなに簡単な操作でカーソルの移動ができると思えば，ぜひ自分のプログラムで応用してやろうと考えるのが人情ですね。

カーソルの移動ができてしまえば，あとに残るのは与えられた文字列（いくつかの文字が格納されているchar型配列）に対する操作だけです。たとえばbuf []という文字列（配列）のposという位置にchという文字を挿入する場合は，

cur_rt = strlen(&buf [pos])；…(1)
for(i=cur_rt;i>=0；i--)
buf[pos+i+1] = buf[pos+i]；…(2)
buf[pos++] = ch; …(3)

というように，
1) カーソルより右にある文字数を調べ
2) カーソル位置を空けるために，カーソルより右にある文字を1文字だけ右に移動し，
3) 空いた位置に文字を挿入する

ことで実現できます。このとき，カーソルの移動は

printf("¥r%s¥r",buf);
for(i=0；i<pos;i++)
putchar(buf [i])；

とすればよいでしょう。printfの書式にある¥rがキャリッジリターンです。

それでは簡単なプログラムを作って，以上のような考えが正しいことを確認しましょう。

リスト4がそのためのプログラムです。文字を1文字ずつ取り出す関数としてDOSライブラリの中のINPOUT関数を利用しています。この関数はキーボードからの入力があればその文字コード，入力がなければ0を返します。キーボード入力の待ち合わせをしない入力関数はINPOUTくらいしかありません。

また，この関数はCTRL+CやCTRL+Dなどの制御文字を返すこともできますから，それらを編集用のコードにすればよく，行編集のためにはまさにうってつけです。

リスト4のプログラムをコンパイルする場合にはdoslib.aまたはdoslib.lをリンクする必要があります。

リスト4

```
 1: #include <stdio.h>
 2: #include <doslib.h>
 3:
 4: unsigned char buf[256];
 5:
 6: void put_line(int pos)
 7: {
 8:     int i;
 9:
10:     printf("¥r%s¥r",buf); /* buf を表示するとともにカーソルを左端に */
11:     for(i=0;i<pos;i++)    /* それからカーソルを pos の位置へ移動    */
12:         putchar(buf[i]);
13: }
14:
15: main()
16: {
17:     unsigned char ch;
18:     int cur_pos;
19:     int cur_right;
20:     int i;
21:
22:     cur_pos = 0; /* カーソル位置を初期化 */
23:     buf[0]  = 0; /* 最初はヌル文字       */
24:     while(1){
25:         while((ch=INPOUT(0xff))==0);      /* キーが押されるのを待つ */
26:         if(ch==3) break;       /* ^C , INPOUT だと割り込みが起きない */
27:         switch(ch){
28:         case  2:/* ^B */
29:             if(cur_pos==0) break;     /* カーソルが左端なら何もしない */
30:             put_line(--cur_pos);      /* カーソル位置を1文字左へ移動し */
31:             break;                    /* 再表示 */
32:         case  6:/* ^F */
33:             if(buf[cur_pos]==0) break;/* カーソルが行末なら何もしない */
34:             putchar(buf[cur_pos++]);  /* 現在の文字を再表示するだけ   */
35:             break;
36:         case 13:/* ¥r */
37:             putchar('¥n');            /* 改行する */
38:             cur_pos = 0;              /* カーソル位置を初期化 */
39:             buf[0]  = 0;              /* ヌル文字        */
40:             break;
41:         default:
42:             if(ch<0x20) break;                    /* 制御文字なら無視する */
43:             cur_right = strlen(&buf[cur_pos]);    /* カーソルより右にある文字数 */
44:             for(i=cur_right;i>=0;i--)             /* カーソル位置を空けるために */
45:                buf[cur_pos+i+1] = buf[cur_pos+i];/* 文字を1文字右に移動する */
46:             buf[cur_pos++] = ch;                  /* カーソル位置に文字を挿入 */
47:             put_line(cur_pos);                    /* 再表示 */
48:         }
49:     }
50: }
```

リスト4では，カーソルの左移動にはCTRL+B，右移動にはCTRL+Fを使用します。それ以外の場合は，制御文字を除き，カーソル位置に入力した文字を挿入します。この時点で漢字コードの入力にはまだ対応していません。

ファイル名の補完

ファイル名の補完は，コマンドラインから入力される単語のほとんどがファイル名であることを考慮し，ファイル名の先頭の2，3の文字を与えるだけで目的のファイル名を導き出そうという機能です。この機能はカレントディレクトリにあるすべてのファイルの名前を知ることができれば80%完成したも同然です。

あとはカーソル位置の左側に接している単語（ファイル名の一部分）で始まる名前のファイルを検索し，候補が複数あればもとの単語を候補の中で最大限に共通な部分ファイル名に拡張します。候補がひとつならばそのファイル名で元の単語を置き換えます。候補がひとつもなければなにもしません。たとえば，

a

という単語を補完することを考えます。いま，カレントディレクトリの中でaで始まるファイルが，

abcd
abce
abcf

の3つであるなら，コマンドラインのaという単語を3つのファイルに共通な，

abc

という単語に置き換えるとともに，候補となるファイルの一覧を画面上にプリントします。

このような処理も，ファイル名の一覧があらかじめわかっていれば，難しくありませんね。そこで，カレントディレクトリのファイル名の一覧を返す関数がないかとマニュアルを探すことになります。

もし，適当なライブラリ関数がなければディスクのディレクトリ領域を調べるプログラムを作ればいいや（口でいうのは簡単だね）と思っていました。しかし，「file」という単語を頼りに探したところ，これまたDOSライブラリのFILES関数とNFILES関数を使用すればいいということがわかりました。これらは，検索するファイル名をワイルドカードで指定できるので，まさにいまの目的にぴったりの関数です。マニュアルにプログラム例も載っていますが，動

作の理解を深めるため自分でサンプルプログラムを書いてみましょう。それがリスト5です。

リスト5では，与えられたディレクトリを示す文字列に対して，

¥*.*

という文字列を付加してFILES関数とNFILES関数を呼び出し，ひとつのファイルが求まるごとにそれをプリントしています。

これらの関数を実際のシェルで使用するためには，与えられた単語に対して，

.

あるいは

*

という文字列を付加してそれらの関数を呼び出すことになるでしょう。

例によって，DOSライブラリを使用するため，リスト5をコンパイルするためにはdoslib.aやdoslib.lをリンクする必要があります。しかし，マニュアルをよく探していく（移植性を考えるとDOS=Human68kに依存する関数はできるだけ使用したくないので）と標準ライブラリ（clib.aまたはclib.l）の中にfiles関数とnfiles関数を見つけることができます。これらの関数は引数の与え方が異なるだけで，効果はFILES関数やNFILES関数と同じようです。最終的なシェルプログラムではfiles関数とnfilesを使用しています。

ところで，候補となるファイルが複数存在する場合は，そのファイル一覧のプリントの方法も問題です。できることなら画面いっぱいにファイル名をプリントしたいものです。これは1行にいくつのファイル名をプリントしたらよいかという問題に換言できるでしょう。

すなわち，表示をバランスよく行うためには，ファイル名の長さの最長のものを求め，それを画面の1行に表示できる文字数に対して割り算を行えば，1行に表示するファイル名の個数がわかります。実際には，ファイル名とファイル名の間に空白を入れて表示する必要があるので，1行に表示できる文字数をファイル名の文字数の最大値＋1で割り算して1行に表示するファイル名の個数を求めます。

このときひとつのファイル名が占める桁数もファイル名の最大値＋1でよいことがわかります。そして，1行に表示できる文字がwidth文字である，count個のファイル名が文字列の配列files[]に格納され，そのファイル名の最大値がmax_lenとすると，

```
    max_len++;   /* 最大値＋1 */
sprintf(fmt,"%%-%ds",max_len);
    /* printf 用の書式を作成 */
width = width/max_len;
    /* 1行のファイル名の個数 */
printf("¥n");
for(i=0; i<count;i++) {
        /* 順番に表示 */
    printf(fmt,files[i]);
    if((i%width)==(width-1))
        printf("¥n");
}
if(i%width) printf("¥n");
    /* 最後が改行でないときは改行 */
```

というプログラムで目的を達することができます。

これは，よく見ると，「配列に格納されているいくつかの要素を1行にn個ずつ表示せよ」という例題の解答と同じものです。ちょっと高度なのは，プログラム内でプリントする桁数を示す書式を作り出している点でしょうか。

このような小プログラムは皆さんが，おりに触れ，C言語の練習問題として経験しているものでしょう。その経験がこのようなところで生かされてくるのです。

さて，ファイル名のプリントでもうひとつ重要なのは1行に表示可能な文字数をどのようにして求めるかということです。多くの人は，そんなの96文字に決まっているというかもしれません。しかし，これは画面モードが768×512ドットのときにいえることであって，必ずしも正しいとは限りません。

それでは，任意の画面モードにおける1行の文字数をどうすれば計算することができるのでしょう。答えは現在の画面モードを知ることです。画面モードは表示画面の横×縦のドット数を一意に決定します。たとえば画面モードが12（IOCSコールにおいて）ならば，その表示画面のサイズは512×512ドットですから，1行の表示可能文字数は，

512/8

で64文字ということになります。このときフォント（半角文字）のサイズは通常の8×16ドットを仮定しています。

画面モードを知る方法にはDOSコールのC_WIDTH関数を使う方法とIOCSコールのCRTMOD関数を使う方法があるようです そのためだけの目的ならどちらも同じような機能ですが，今回はCRTMOD関数のほうを使用してみました（単にマニュアルを探しているときに最初に見つけただけ）。

コマンド名の補完

コマンド名の補完はコマンドラインの最初（いちばん左側）の単語に対して行うのが普通です。なぜならばそれがコマンド名（の一部）であることは明らかだからです。

この処理のキーポイントを考えてみましょう。このとき，ほとんどの処理はファイル名の補完と同じになります。唯一の違いはファイル名の検索対象ディレクトリが，カレントディレクトリであるか，環境変数PATH（あるいはpath）の中で指定されたどこかほかのディレクトリであるかということです。

PATHに含まれるディレクトリはひとつではありませんから，そのディレクトリを順次切り出しては，それに実行形式のファイルの拡張子を考慮した，

*.X
*.Z
*.R

リスト5

```
 1: #include <stdio.h>
 2: #include <doslib.h>
 3:
 4: char line[255];
 5:
 6: main()
 7: {
 8:     unsigned char   dir[100];
 9:     struct   FILBUF fbuf;
10:     int cnt;
11:
12:     cnt = 0;
13:     while(1){
14:         printf("ディレクトリ=>");
15:         if(gets(line)==NULL) break;
16:         sscanf(line,"%s",dir);
17:         strcat(dir,"¥¥*.*");
18:         if(FILES(&fbuf,dir,0x3f)>=0){             /* 一致するファイル？ */
19:             printf("(%d)%s¥n",cnt++,fbuf.name);/* あったら表示       */
20:             while(NFILES(&fbuf)>=0)            /* 残りがあるだけ     */
21:             printf("(%d)%s¥n",cnt++,fbuf.name);/* それも表示         */
22:         }
23:     }
24: }
25:
26:
```

＊.BAT
という文字列を結合したファイル名（ワイルドカード指定）でFILES関数やNFILES関数を呼び出すという操作を繰り返します。FILES関数とNFILES関数の使い方については，ファイル名の補完で学習したはずなので，結局キーポイントは環境変数PATHの中から，ディレクトリを次々と取り出していく処理になります。

環境変数を取り出すためにＣ言語のライブラリではgetenv関数が用意されています。これも，なにか値を得るための関数の接頭語としてよく使われるgetと,「環境」が英語で「environment」になることを考慮してマニュアルを探せば，比較的簡単に見つけることができるでしょう（さっきから適当なライブラリを探すためには英語力を身につけろといっているように思えるな)。

PATHという環境変数はいくつかのディレクトリが「；」で区切られている文字列です。ここでは，実験として，環境変数PATH（あるいはpath）に含まれるディレクトリを，順次切り出してプリントするプログラムを作ってみましょう。それがリスト６です。リスト６ではpathというchar型配列にひとつのディレクトリを格納しては，すぐさまprintf関数でプリントしています。

ところで，一般に，コマンド名の補完はコマンドラインの最初の単語に対して行われますが，今回のシェルでは単語がちょうどコマンドラインの左端から始まっている場合のみ，コマンド名の補完を行うようにしています。たとえそれが最初の単語であっても，その左に空白やタブがあるときには，単にカレントファイルのファイル名で補完を行うようにします（実行形式か否かの拡張子の判別はしません)。新しいプログラムを開発しているときなど，カレントディレクトリ内だけのファイルを参照したい場合を考慮しています（単なる趣味ですけど)。

標準入出力のリダイレクト

標準入出力のリダイレクトは今回もっとも悩んだ項目です。いま行いたいことは，コマンドラインが，

prog < infile > outfile

のようになっている場合に，

1) 標準入力をinfileに切り替える
2) 標準出力をoutfileに切り替える
3) progというコマンドを起動する
4) 標準入力と標準出力を元に戻す

という処理をすることです。

しかし，私はファイルの切り替え（リダイレクト）のために行う具体的なプログラム処理を考えつくことができませんでした。Ｃ言語のライブラリでは標準入出力のようにあらかじめオープンされているストリームをファイルに割り当て直すために，

freopen

関数が用意されています。当初，２～３の簡単なプログラムによる実験ではfreopen関数を使えば，すべてうまくいくように思えました。たとえば，freopen関数で標準出力をfooというファイルに切り替える(リダイレクトする）ことを考えると次のようになります。

```
printf("Hello, Usako.¥n");
fp=freopen("foo","w",stdin);
printf("How are you?¥n");
fclose(fp);
printf("I am very fine.¥n");
```

このとき，freopenからfcloseまでの間に行われた標準出力への出力である，

"How are you?¥n"

という文字列はfooというファイル内に書き込まれ，それ以外も文字列は画面上に表示されます。

しかし，この方法には重大な欠点がありました。freopen関数による標準出力の切り替えは子プロセスには影響を及ぼさないのです。

たとえば，

```
fp=freopen("foo","w",stdin);
spawnv(P_WAIT,argv[0],argv);
fclose(fp);
```

のようなプログラムでは，spawnvで起動されるプログラムが標準出力に書き込みを行うと，それはfooというファイルに書き込まれず，画面上に表示されてしまいます。これではシェルの目的と合致しません。ここで私は途方にくれてしまったのです。

さて，このとき私が取った行動は次のどれでしょう。

1) COMMAND.XやFISH.Xなどリダイレクトをサポートしているコマンドシェルを逆アセンブルして，リダイレクトの機構がどうなっているのか調べた。
2) 本屋の棚に並んでいるＣ言語の参考書を片っ端から流し読みし，標準入出力のリダイレクトがプログラム例として採用されているものがないか探した。
3) C MAGAGINE（ソフトバンク刊）のバックナンバーを読みあさり，リダイレクトに関する記事はないか探した。

答えは1)～3)のすべてです。ただ，3)はテーマの検索にやたら時間が掛かるので早々に諦めました。1)はいわゆるリバースエンジニアリングというやつです。逆アセンブルリストの中で，コマンドラインの文字に関し，'<'や'>'の文字コードで比較を行っている部分を見つけ，そこから処理の流れを追っていきます。

それにしても，他人の書いたプログラムを読むのは非常にためになります。ただ，COMMAND.XやFISH.Xは複数のプログラムがバインドされている（しかも不可視属性で）のでそのままではDIS.Xで逆アセンブルできません。ちょっとした工夫が必要です。2)に関しては１冊だけリダイレクトを扱ったもの（書名は忘れた）があったので必死に記憶して帰りました。結局，1)と2)の行動から次のことを学びました。

リスト6

```
 1: #include <stdio.h>
 2:
 3: main()
 4: {
 5:     char *sta,*end;
 6:     char path[90];
 7:     int  cnt;
 8:
 9:     if((sta=(char*)getenv("PATH"))==NULL){
10:         sta=(char*)getenv("path");
11:     }
12:     if(sta==NULL){
13:         fprintf(stderr,"環境変数 PATH がない¥n");
14:         exit(1);
15:     }
16:     end = sta;
17:     cnt = 0;
18:     while(*end){
19:         if(*end==';'){          /* ディレクトリは ; で区切られている */
20:             strncpy(path,sta,end-sta);   /* ; の直前までをコピー */
21:             path[end-sta] = 0;           /* 文字列の終わり      */
22:             printf("(%d)%s¥n",cnt,path); /* 名前を表示          */
23:             cnt++;
24:             sta = end+1;                 /* ; を飛ばす */
25:         }
26:         end++;
27:     }
28:     strncpy(path,sta,end-sta);           /* 最後のパス名をコピー */
29:     path[end-sta] = 0;                   /* 文字列の終わり      */
30:     printf("(%d)%s¥n",cnt,path);         /* 名前を表示          */
31:     exit(0);
32: }
```

「dup関数とdup2関数を使えばよい」
具体的には以下の手順で処理します。

1) 元の標準入出力のファイルハンドルをdup関数で変数にコピーしておく。
2) 切り替え対象のファイルをオープンする。
3) オープンしたファイルのファイルハンドルをdup2関数で標準入出力のファイルハンドルにコピーする（これで切り替えが終了する）
4) なにか適当な処理をする。
5) コピーしておいたファイルハンドルをdup2関数で標準入出力のファイルハンドルにコピーする（これで切り替えが元に戻る）。

ここで，ファイルハンドルという，普段はfopenという高レベルの関数を使ってファイルをオープンしている人には，聞き慣れない単語が出てきました。ファイルハンドルとはopenという低レベルの関数でファイルをオープンするときに返されるファイルの管理番号です。fopen関数はFILE構造体へのポインタであるファイルポインタを値として返してきますが，ファイルハンドルとはファイルポインタの基礎になるものと思っておけばよいでしょう。

ファイルハンドルを使用するため，リダイレクトを行うためには，あまり使い慣れないopen関数を使わなければならないでしょうか。実はそうではありません。ファイルハンドルはファイルポインタからfileno関数で取り出すことができます。したがって，ファイルのオープンはいつものとおりfopen関数で行えばよいのです。

試しにサンプルプログラムを作って動作を確かめましょう。それがリスト7です。リスト7は子プロセスが標準出力に書き込む文字列をresult.outというファイルにアペンドしていくプログラムです。そして，いま子プロセスとして起動されるhello.xというコマンドは，C言語を知っている人にはあまりにも有名な，

```
main()
  {
      printf("Hello, world.¥n");
  }
```

というプログラムをコンパイルしてできるコマンドです。

リスト7はかなり短いプログラムですがここまでくるためにはかなりの時間を費やしてしまいました。しかし，これで子プロセスにおける標準入出力の切り替えができるようになりました。ひとつ利口になったような気分です。

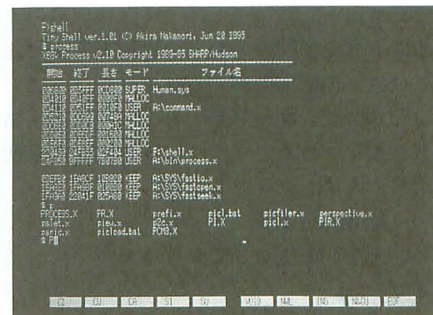
そしてシェルの完成

以上でシェルの作成に必要な機能ブロックの考察が終わりました。あとはそれを適当に変更しながら組み合わせていくだけの作業です。実際には，私は上述した実験をすべて行ってから目的であるシェルのプログラムを書き始めたわけではありません。考察をしながら並行して目的のプログラムを書いていき，よくわからない部分に行き当たったら小さなプログラムを作って実験する，という手法をとりました。

もっとも原始的なプログラムはリスト2です。それに対して次々と新しい機能を加えていって最終的な形に仕上げたのです。それがリスト8です。リスト1～7のプログラムがリスト8の中のどの部分に生かされているか探してみましょう。

ところで，リスト8の中ではコマンドラインの入力に漢字コード（シフトJIS）が与えられても（それなりに）動作するように作ってあります。リスト中の改版履歴を見てもらえばわかりますが，標準入出力のリダイレクトができるようになるまでが約10日です。

その後，漢字コードの入力に対応するためには倍の時間がかかっています。やはり2バイトコードの扱いは難しいですね。かなり四苦八苦しました（行編集のやり方が少し変更になった）が，どの部分かはリストから想像してみてください。結構付け焼き刃的な記述が目につくと思います。一応動作するということですっきりとした記述に書き直すことはしていません。

そうそう，リスト8のコマンドシェルのキーバインドを簡単に説明しておかねばなりません。それは以下のようになっています。

完成したコマンドシェル

- カーソルを1文字右　CTRL+F
- カーソルを1文字左　CTRL+B
- カーソル位置の1文字を削除 CTRL+D
- カーソルの左の1文字を削除 CTRL+H
- カーソルを行の先頭に移動　CTRL+A
- カーソルを行の最後に移動　CTRL+E
- 前のコマンドラインを呼び出すCTRL+P
- 後のコマンドラインを呼び出すCTRL+N
- コマンド名/ファイル名の補完　TAB
  またはESC2回

残念ながら，現段階ではこれらの機能を自由なキーに割り当てることはできません。キーバインドができるように改造するのはなかなか興味深い練習問題でしょう。

あと，忘れていましたが，このシェルを終了するための組み込みコマンドは，

exit

です。EOF(CTRL+Z)では終了できないので注意しましょう。

ところで，リスト8のプログラムはXCのライブラリを使っている限りは正しく動作すると思いますが，『X68k Programming

リスト7

```
 1: #include <stdio.h>
 2: #include <process.h>
 3: #include <io.h>
 4:
 5: char *std_out="result.out";
 6: char *argv[]={"hello.x",NULL}; /* コプロセスで起動するファイル名 */
 7:
 8: int  fhdl;  /* 標準出力記憶用のファイルハンドル */
 9: FILE *fptr; /* 標準出力切り替え用のファイルポインタ */
10:
11: main()
12: {
13:     fhdl = dup(fileno(stdout));/* stdout のファイルハンドルを記憶  */
14:     fptr = fopen(std_out,"a"); /* アペンドモードでオープンしてみる */
15:     if(fptr==NULL){
16:         fprintf(stderr,"ファイルがオープンできません。¥n");
17:         exit(1);
18:     }
19:     fseek(fptr,0,SEEK_END);     /* ファイルの最後にシーク */
20:     if(dup2(fileno(fptr),fileno(stdout))<0){ /* リダイレクト！ */
21:         fprintf(stderr,"ファイルがリダイレクトできません。¥n");
22:         exit(1);
23:     }
24: /* ここから標準出力に書き込むと指定したファイルに入る */
25:     spawnv(P_WAIT,argv[0],(const char**)argv); /* 子プロセス起動 */
26: /* ここまで */
27:     fclose(fptr);
28:     dup2(fhdl,fileno(stdout)); /* stdout をもとに戻す */
29:     exit(0);
30: }
```

Series libc』（ソフトバンク刊）に収録のライブラリでは期待どおりの動作をしません。これは，標準出力が，バッファがいっぱいになるか改行コードがくるまで画面上に出力されないためです。このライブラリを使用する人は，コメントで殺している，

#define LIBC

という1行を生かしてからコンパイルしてください。

おわりに

今回は私がプログラムを作る場合において，ものごとを考えていく過程をほとんどそのまま示してみました。いかがだったでしょう。プログラミングにおいては，ある機能を実現するために，どのようなことをしたらよいかを考えつくことがもっとも重要であるということが伝わったでしょうか。

さて，どのようなことをいってみても最終的にはどのくらい場数を踏んだかが問題になります。ある機能をプログラムで実現しようとするとき，過去に経験（見たり書いたり）したプログラム例が応用できる場合が非常に多いからです。そのようなことを常に心がけながら，皆さんもどんどんプログラムを作っていってください。

リスト8

```
  1: #define VERSION 0x101
  2: #define STANDARD
  3: /* #define LIBC        /* LIBC を使用するとき定義する */
  4: /*
  5:     tiny shell --- 中森  章
  6:
  7:      [改版履歴]
  8:
  9:       ver 0.01 05/22/1993 新規作成
 10:                           コマンドラインの編集(^B,^F,^H,^P,^N)機能
 11:                           組み込み命令は exit のみ
 12:       ver 0.02 05/23/1993 ファイル名の補完機能を追加
 13:       ver 0.02 05/24/1993 コマンド名の補完機能を追加
 14:                           おや？ .BAT は spawnv では実行できないぞ
 15:       ver 0.03 05/25/1993 ということで、拡張子が BAT のファイルは
 16:                           COMMAND.X で実行するようにした
 17:       ver 0.03 05/27/1993 やっぱりカーソル位置の文字を消去できなければ
 18:                           いけないからね
 19:       ver 0.04 05/30/1993 標準入出力のリダイレクトをサポート
 20:       ver 0.05 06/12/1993 漢字入力に一応対応
 21:       ver 0.06 06/13/1993 'リ'などのように2バイト目が 0x80 以上の漢字コード
 22:                           があるとき'りa'という文字列から1文字消去すると'a'
 23:                           を全角文字の2バイト目と思ってしまうバグに対処
 24:       ver 0.06 06/14/1993 calc_prev で 2 の位置が与えられたとき無条件に0の
 25:                           位置が返るバグをフィックス
 26:       ver 1.00 06/19/1993 複数に分かれていたファイルをOh!Xに掲載するために
 27:                           ひとつにまとめた。たくさんあるとうっとおしいので。
 28:                           ついでに LIBC でコンパイルしても動作するよう改造。
 29:                           LIBC では 10K バイト程度実行ファイルが大きくなる！
 30:       ver 1.01 06/20/1993 組み込みコマンドを実行するときにCOMMAND.Xのコプロ
 31:                           セスとして立ち上げることを止めた。
 32:
 33:      [問題点]
 34:          1)   文字入力が1行を越えると再表示が1行下のラインになってしまう
 35: */
 36: #include <stdio.h>
 37: #include <process.h>
 38: #include <doslib.h>
 39: #include <iocslib.h>
 40: #include <io.h>
 41:
 42: #ifdef LIBC
 43: #include <unistd.h>
 44: #include <string.h>
 45: typedef unsigned char UBYTE;
 46: #undef  STANDARD
 47: #include <sys/dos.h>
 48: #define putchar(x)    putchar(x),fflush(stdout)
 49: #endif
 50:
 51: #define MAX_HISTORY 10
 52: #define MAX_LINE    512
 53:
 54: static char version[]="Tiny Shell ver.%3.2f (C) Akira Nakamori, " __DATE__ "¥n";
 55: static int quit_flag;
 56: static int cmd_status;
 57: static char prompt[MAX_LINE]={'$',' ',0};
 58:
 59: static char cmd_line[MAX_LINE];
 60: #if VERSION>=0x004
 61: int  cmd_argc;
 62: char *cmd_argv[MAX_LINE];
 63: #else
 64: static int  cmd_argc;
 65: static char *cmd_argv[MAX_LINE];
 66: #endif
 67: static char cmd_history[MAX_HISTORY][MAX_LINE];
 68: static int  history_ptr;
 69: static int  history_pend;
 70:
 71: /* 外部定義関数のプロトタイプ */
 72: /* ファイルを1つにまとめたので、もはや「外部定義」ではないが */
 73:
 74: void cd_cmd (char *dir_name);
 75: int  dos_cmd(char *args[]);
 76: int  complete(char buf[],int cur_pos);
 77: int  cmd_complete(char head[],char files[][25]);
 78: int  crt_width(void);
 79: int  redirect(void);
 80: void reredirect(void);
 81:
 82: /*
 83:  * いろいろな変数の初期化
 84: */
 85: void setup(void)
 86: {
 87:     int i;
 88:
 89:     quit_flag   = 0;  /* 終了するか否かを示す */
 90:     cmd_status  = 0;  /* spawnv の終了コード */
 91:     history_ptr  = 0; /* ヒストリで、次にコマンドを格納する場所     */
 92:     history_pend = -1; /* 現在表示しているコマンドが格納されている場所 */
 93:     for(i=0;i<MAX_HISTORY;i++) /* ヒストリーを初期化 */
 94:         cmd_history[i][0] = 0;
 95:     printf(version,(double)VERSION/256.0+
 96:                   (double)(VERSION%256)/100.0); /* バージョン */
 97: }
 98:
 99: /*
100:  * 文字列を画面に表示し直す
101:  * カーソル位置を文字列の終わりに移動する機能もある
102: */
103: void putstr(char *str)
104: {
105:     while(*str)
106:         putchar(*str++);
107: }
108:
109: /*
110:  * 元の文字の代わりに空白を表示する
111: */
112: void era_line(char *buf)
113: {
114:     putchar('¥r');      /* カーソルを左端へ      */
115:     while(*buf++)       /* 文字数だけ空白を表示 */
116:         putchar(' ');
117:     putchar(' ');       /* ちょっと余分に空白を */
118:     putchar(' ');
119:     putchar('¥r');      /* カーソルを左端へ      */
120: }
121:
122: /*
123:  * コマンドラインを表示し直す
124: */
125: void put_line(char buf[],int pos)
126: {
127:     int i;
128:
129: #if 0
130:     putchar('¥r');   /* 同じことは printf でもできる */
131:     putstr (prompt); /* プロンプトを表示             */
132:     putstr (buf);    /* コマンドラインの文字列を表示 */
133:     putchar(' ');    /* バックスペース用             */
134:     putchar('¥r');   /* いったんカーソルを左端へ     */
135:     putstr (prompt); /* カーソルをプロンプトの右端へ */
136: #else
137:     /* printf で実現 */
138: #if VERSION>=0x005
139:     printf("¥r%s%s  ¥r%s",prompt,buf,prompt);/* 全角文字を考慮して空白2つ */
140: #else
141:     printf("¥r%s%s ¥r%s",prompt,buf,prompt);
142: #endif
143: #endif
144:     for(i=0;i<pos;i++)  /* カソールを pos の位置へ移動 */
145:         putchar(buf[i]);
146: #ifdef LIBC
147:     fflush(stdout);
148: #endif
149: }
150:
151: #if VERSION>=0x006
152: /*
153:  *  与えられた位置よりも1つ前の文字位置を返す
154: */
155: int calc_prev(char buf[],int pos)
156: {
157:     unsigned char *p;
158:     int nxt,nxt1,ch;
159:
160:     if(pos<2)
161:         return (0);
162:     p = (unsigned char*)buf;
163:     ch = p[0];
164:     if((ch>=0x80 && ch<0xa0)||(ch>=0xe0)){  /* 漢字コードの1バイト目 */
165:         if(pos==2) return (0);
166:         nxt = 2;
167:     }
168:     else{
169:         if(pos==2) return (1);
170:         nxt = 1;
171:     }
172:     for(;nxt<pos;nxt++){
173:         nxt1 = nxt;
174:         ch = p[nxt];
175:         if((ch>=0x80 && ch<0xa0)||(ch>=0xe0))  /* 漢字コードの1バイト目 */
176:             nxt++;
177:     }
178:     return (nxt1);
179: }
180: #endif
181: /*
182:  * コマンドラインを得る
183:  * CR が押されるまでコマンドラインを編集できる
184:  * ヒストリ機能をサポートする
185: */
186: void get_line(char buf[])
187: {
188:     int  ch,i,ch2;
189:     int  cur_pos=0;  /* 現在のカーソル位置                */
190:     int  cur_right;  /* カーソル位置から行末までの文字数  */
191:     int  cr_flag=0;  /* CR キーが押されたら1になる        */
192:     int  prefix=0;   /* 1回目の ESC を記憶するためのフラグ */
193:
194:     buf[cur_pos] = 0; /* 最初は文字は入力されていない */
195:     while(!cr_flag){
196:         switch(ch=INPOUT(0xff)){ /* キーバッファから1文字取り出す */
197:         case 0:/* 入力なし → 何もしない */
198:                 break;
199:         case 1:/* ^A → カーソルを左端へ */
200:                 cur_pos = 0;                /* カーソル位置を左端にして */
201:                 put_line(buf,cur_pos);      /* 再表示 */
202:                 break;
203:         case 2:/* ^B → カーソルを1文字左へ */
204:                 if(cur_pos==0) break;       /* カーソルが左端なら何もしない */
205: #if VERSION>=0x005
206: #if VERSION>=0x006
207:                 cur_pos = calc_prev(buf,cur_pos)+1;
208: #else
209:                 ch = buf[cur_pos-2]&0xff;  /* 2カラム前の文字 */
210:                 if((cur_pos>1)&&           /* 漢字が1カラム目のはずがない */
211:                    ((ch>=0x80 && ch<0xa0)||(ch>=0xe0)) ) /*漢字コードの1バイト目*/
212:                     cur_pos--;              /* いったん1文字左へ */\
213: #endif
214: #endif
215:                 put_line(buf,--cur_pos);   /* カーソル位置を1文字左へ移動し */
216:                 break;                     /* 再表示 */
217:         case 4:/* ^D → カーソル位置の文字を消去 */
218:         case 0x7f:/* ^? */
219:                 if(buf[cur_pos]==0) break; /* カーソルが行末なら何もしない */
220:                 i = cur_pos;
```

```
221: #if VERSION>=0x005
222:                 ch = buf[i]&0xff;          /* カーソル位置の文字 */
223:                 if((ch>=0x80 && ch<0xa0)||(ch>=0xe0)) /*漢字コードの1バイト目*/
224:                     while(buf[i]=buf[i+2]) i++;       /*文字を2文字左へ移動 */
225:                 else                                  /*ちょっと変な記述だけど*/
226: #endif
227:                 while(buf[i]=buf[i+1]) i++;/* 文字を1文字左へ移動 */
228:                 put_line(buf,cur_pos);     /* 再表示 */
229:                 break;
230:         case 5:/* ^E → カーソルを行末へ */
231:                 cur_pos = strlen(buf);     /* 行末(右端)を求めて */
232:                 put_line(buf,cur_pos);     /* 再表示 */
233:                 break;
234:         case 6:/* ^F → カーソルを1文字右へ */
235:                 if(buf[cur_pos]==0) break; /* カーソルが行末なら何もしない */
236: #if VERSION>=0x005
237:                 ch = buf[cur_pos]&0xff;        /* カーソル位置の文字 */
238:                 if((ch>=0x80 && ch<0xa0)||(ch>=0xe0)){ /*漢字コードの1バイト目*/
239:                     ch2 = buf[cur_pos+1]&0xff; /* 2文字目 */
240:                     putchar(ch);               /* 再表示  */
241:                     putchar(ch2);
242:                     cur_pos += 2;
243:                     break;
244:                 }
245: #endif
246:                 putchar(buf[cur_pos++]);   /* 現在の文字を再表示するだけ*/
247:                 break;
248:         case 8:/* ^H → カーソルの1文字左の文字を消去 */
249:                 if(cur_pos==0) break;      /* カーソルが左端なら何もしない*/
250:                 i = cur_pos;
251: #if VERSION>=0x006
252:                 cur_pos = calc_prev(buf,cur_pos); /*1文字左のカーソル位置*/
253:                 ch = i-cur_pos;                   /*何文字分移動するか  */
254:                 while(buf[i-ch]=buf[i]) i++;      /*文字をそれだけ左へ移動*/
255: #else
256: #if VERSION>=0x005
257:                 ch = buf[i-2]&0xff;
258:                 if( (i>1)&&
259:                     ((ch>=0x80 && ch<0xa0)||(ch>=0xe0))){/*漢字コードの1バイト目*/
260:                     while(buf[i-2]=buf[i]) i++;         /*文字を2文字左へ移動*/
261:                     cur_pos--;
262:                 }
263:                 else                              /* ちょっと変な記述だけど */
264: #endif
265:                 while(buf[i-1]=buf[i]) i++;/* 文字を1文字左へ移動 */
266:                 cur_pos--;                 /* ^Dとの違いに注意 */
267: #endif    /* VERSION>=0x006 */
268:                 put_line(buf,cur_pos);     /* 再表示 */
269:                 break;
270:         case 9:/* TAB → コマンド/ファイル名の補完 */
271:                 cur_pos = complete(buf,cur_pos);
272:                 put_line(buf,cur_pos);
273:                 break;
274:         case '¥n':/* リターン → 呼び出し元へ戻る */
275:         case '¥r':
276:                 putchar('¥n');
277:                 cr_flag = 1;
278:                 break;
279:         case ('n'-'a'+1):/* ^N → ひとつ後に入力されたコマンドを表示 */
280:                 if(history_pend>=0)
281:                    i = history_pend;
282:                 else
283:                    i = 0;
284:                 era_line(buf);
285:                 strcpy(buf,cmd_history[i]);
286:                 cur_pos = strlen(buf);
287:                 put_line(buf,cur_pos);
288:                 i++;
289:                 history_pend = (i>=MAX_HISTORY)? 0 : i;
290:                 break;
291:         case ('p'-'a'+1):/* ^P → ひとつ前に入力したコマンドを表示 */
292:                 if(history_pend>=0)
293:                    i = history_pend;
294:                 else
295:                    i = history_ptr-1;
296:                 era_line(buf);
297:                 strcpy(buf,cmd_history[i]);
298:                 cur_pos = strlen(buf);
299:                 put_line(buf,cur_pos);
300:                 i--;
301:                 history_pend = (i<0)? (MAX_HISTORY-1) : i;
302:                 break;
303:         case 0x1b:/* ESC */
304:                 if(prefix){ /* ESC が2回押された → TAB と同じ動作 */
305:                     prefix = 0;
306:                     cur_pos = complete(buf,cur_pos);
307:                     put_line(buf,cur_pos);
308:                 }
309:                 else        /* ESC の1回目 */
310:                     prefix = 1;
311:                 break;
312:         default: /* 文字を現在のカーソル位置へ挿入する */
313:                 if(ch<0x20) break;                 /*制御文字なら無視する     */
314:                 cur_right = strlen(&buf[cur_pos]); /*カーソルより右にある文字数*/
315: #if VERSION>=0x005
316:                 if((ch>=0x80 && ch<0xa0)||(ch>=0xe0)){ /*漢字コードの1バイト目*/
317:                     ch2 = INPOUT(0xff);                /*2バイト目もあるよね  */
318:                     for(i=cur_right;i>=0;i--)
319:                         buf[cur_pos+i+2] = buf[cur_pos+i]; /*文字を2文字右に移動する*
/
320:                         buf[cur_pos++] = ch;
321:                         buf[cur_pos++] = ch2;
322:                         put_line(buf,cur_pos);          /* 再表示 */
323:                         break;
324:                 }
325: #endif
326:                 for(i=cur_right;i>=0;i--)          /* カーソル位置を空けるために */
327:                    buf[cur_pos+i+1] = buf[cur_pos+i]; /* 文字を1文字右に移動する */
328:                 buf[cur_pos++] = ch;                /* カーソル位置に文字を挿入 */
329:                 put_line(buf,cur_pos);              /* 再表示 */
330:         }
331:         if(ch!=0 && prefix && ch!=0x1b) /* ESC 以外のキーなら、先の ESC を忘れる */
332:             prefix = 0;
333:     }
334: }
335:
336: /*
337:  * 1行を入力する
338: */
339: void get_cmd(void)
340: {
341:     char *bgn,*end;
342:
343:     printf("%s",prompt);/* プロンプトを表示 */
344: #ifdef LIBC
345:     fflush(stdout);
346: #endif
347:     get_line(cmd_line); /* 1行入力      */
348:
349:     if(cmd_line[0]!=0){ /* 入力があればヒストリに登録 */
350:         strcpy(cmd_history[history_ptr++],cmd_line);
351:         if(history_ptr>=MAX_HISTORY)
352:             history_ptr = 0;
353:         history_pend = -1;
354:     }
355:
356: /* 以下は入力されたコマンドラインを単語に切り分ける処理 */
357:     bgn = cmd_line;
358:     cmd_argc = 0;
359:     while(1){
360:         while(  (*bgn==' ' )
361:                ||(*bgn=='¥t')) bgn++;
362:         end = bgn;
363:         if(  (*bgn==0   )
364:            ||(*bgn=='¥n')) break;
365:         while(  (*end!=' ' )
366:                &&(*end!='¥t')
367:                &&(*end!='¥n')
368:                &&(*end!='¥r')
369:                &&(*end     )) end++;
370:         cmd_argv[cmd_argc++] = bgn;
371:         if(*end==0) break;
372:         *end = 0;
373:         bgn = ++end;
374:     }
375:     *end = 0;
376:     cmd_argv[cmd_argc] = 0;
377: }
378:
379: /*
380:  * 組み込みコマンドかどうかを調べ
381:  * そうなら組み込みコマンドを実行する
382:  * そうでなければ何もしない
383: */
384: int do_builtin(void)
385: {
386:     int status;
387:
388:     if(strcmp(cmd_argv[0],"exit")==0){
389:         quit_flag = 1;
390:         return (1);
391:     }
392:     if(strcmpi(cmd_argv[0],"cd")==0){
393:         cd_cmd(cmd_argv[1]);
394:         return (1);
395:     }
396:     if(  (strcmpi(cmd_argv[0],"dir")==0)
397:        ||(strcmpi(cmd_argv[0],"copy")==0)
398:        ||(strcmpi(cmd_argv[0],"del")==0)
399:        ||(strcmpi(cmd_argv[0],"type")==0)
400:        ||(strcmpi(cmd_argv[0],"screen")==0)
401:        ||(strcmpi(cmd_argv[0],"md")==0)
402:        ||(strcmpi(cmd_argv[0],"rd")==0)
403:     ){
404:         (void)dos_cmd(&cmd_argv[0]);
405:         return (1);
406:     }
407:     return (0);
408: }
409:
410: /*
411:  * ファイル名の拡張子が BAT であるかを調べる
412: */
413: int chk_bat(char name[])
414: {
415:     int len;
416:     len = strlen(name);
417:     if(len<4) return (0);
418:     return ((        name[len-4] =='.')
419:            &&(toupper(name[len-3])=='B')
420:            &&(toupper(name[len-2])=='A')
421:            &&(toupper(name[len-1])=='T'));
422: }
423:
424: /*
425:  * コマンドを実行する
426: */
427: void exec_cmd(void)
428: {
429:     if(cmd_argv[0]==0) /* 入力なし */
430:         return ;
431:
432:     if(do_builtin())
433:         cmd_status = 0;
434:     else{
435:         if(chk_bat(cmd_argv[0]))
436:             cmd_status = dos_cmd(&cmd_argv[0]);
437:         else
438:             cmd_status = spawnv(P_WAIT,cmd_argv[0],(const char**)cmd_argv);
439:         if(cmd_status<0)
440:             fprintf(stderr,"%s: コマンドが実行できません。¥n",cmd_argv[0]);
441:     }
442: }
443:
444: /*
445:  * メインルーチン
446: */
447: void main(void)
448: {
449:     setup();             /* 変数の初期化 */
450:     while(!quit_flag){
451:         get_cmd();       /* コマンドを得る    */
452: #if VERSION>=0x004
453:         if(redirect()<0)/* リダイレクト要求があればリダイレクトする */
454:            continue;   /* エラーが出ていたら無視 */
455: #endif
456:         exec_cmd();      /* コマンドを実行する */
457: #if VERSION>=0x004
458:         reredirect();    /* リダイレクトをもとに戻す */
459: #endif
460:     }
461:     exit(0);
462: }
463: /************************************************************
464:   組み込みコマンドの処理の実体
465: ************************************************************/
466: void cd_cmd(char *dir_name)
467: {
468:         int  cdrv,drv;
469:         char dir[90];
470:
471:         cdrv = drv = CURDRV();
472:         if(dir_name==0){              /* ディレクトリ指定なし */
473:             CURDIR(0,(UBYTE*)dir);
474:             printf("%c:¥¥%s¥n",cdrv+'A',dir);
475:             return ;
476:         }
477:         if(dir_name[1]==':'){ /* ドライブ変更あり */
478:             drv = dir_name[0];
479:             if(drv>='a' && drv<='z')
480:                 drv = drv-'a';
481:             else
482:                 drv = drv-'A';
483:             dir_name += 2;
484:         }
485:         if(cdrv != drv){
486:             if(CHGDRV(drv)<drv){
487:                 fprintf(stderr,"%c: ドライブ変更ができません。¥n",drv+'A');
488:                 return ;
489:             }
```

```
490:            else{
491:                cdrv = CURDRV();   /* もう一度カレンドドライブを見る */
492:                if(cdrv != drv){
493:                    fprintf(stderr,"%c: ドライブ変更ができません。¥n",drv+'A');
494:                    return ;
495:                }
496:            }
497:        }
498:        if(*dir_name==0)       /* ドライブのみの変更 */
499:            return ;
500:        if(CHDIR((UBYTE*)dir_name)<0)
501:            fprintf(stderr,"%s: ディレクトリ変更ができません。¥n",dir_name);
502:        return ;
503: }
504:
505: int dos_cmd(char *args[])
506: {
507:     char *argv[128];
508:     int  i;
509:
510:     argv[0]="COMMAND.X";
511:     i = 0;
512:     while((i<127)&&(argv[i+1]=args[i])) i++;
513:     return spawnv(P_WAIT,"COMMAND.X",(const char**)argv);
514: }
515:
516: /*******************************************************
517:   ファイル名補完処理の実体
518: ********************************************************/
519: static char word[1024];
520: static char word_cmn[1024];
521: static int  c_count;
522: static char c_files[1024][25];
523:
524: /*
525:  * 画面の横方向の文字数を返す
526: */
527: int crt_width(void)
528: {
529:     switch(CRTMOD(-1)){ /* 現在の画面モード */
530:     case  0:
531:     case  1:
532:     case  4:
533:     case  5:
534:     case  8:
535:     case  9:
536:     case 12:
537:     case 13:
538:             return (512/8);
539:     case  2:
540:     case  3:
541:     case  6:
542:     case  7:
543:     case 10:
544:     case 11:
545:     case 14:
546:     case 15:
547:             return (256/8);
548:     case 16:
549:             return (768/8);
550:     case 17:
551:     case 18:
552:             return (1024/8);
553:   default:
554:             return (768/8); /* もし知らない画面モードなら */
555:     }
556: }
557:
558: /*
559:  * バッファの指定した位置（カーソル位置）にファイル名を挿入して
560:  * 新しいカーソル位置を返す
561: */
562: static int f_insert(char buf[],int pos,int head,char str[])
563: {
564:     int len_b,len_s;
565:     int i;
566:
567:     len_b = strlen(buf);      /* バッファの終わりを得る */
568:     len_s = strlen(str)-head; /* ファイル名の入力されていない文字数 */
569:     for(i=len_b;i>=pos;i--)   /* その文字数だけ */
570:         buf[len_s+i] = buf[i];/* バッファに空きを作り */
571:     strncpy(&buf[pos-head],str,len_s+head);/* ファイル名の残りをコピー */
572:     return (pos+len_s);       /* コピーした文字数だけカーソル位置は増加 */
573: }
574:
575: /*
576:  * c_files[] すべてに共通な文字があればそれを求める
577: */
578: static int f_common(void)
579: {
580:     int i,j;
581:     int ch;
582:     int chu;
583:
584:     for(i=0;i<1024;i++){      /* 最初の文字から終わりの文字まで */
585:         chu = toupper(ch=c_files[0][i]); /* 0 番目の要素と比較 */
586:         for(j=1;j<c_count;j++)
587:             if(chu != (toupper(c_files[j][i]))) break;
588:         if(j<c_count) break; /* 一致しない要素があれば検索を終了する */
589:         word_cmn[i] = ch;    /* すべてが一致すればその文字を記録する */
590:     }
591:     word_cmn[i] = 0; /* 文字列を終わらせる */
592:     return (i);
593: }
594:
595: /*
596:  * c_files[] の内容を画面一杯に表示する
597: */
598: static void f_print(void)
599: {
600:     int  i;
601:     int  width;
602:     int  max_len,tmp_len;
603:     char fmt[10];
604:
605:     width = crt_width()-1;  /* 画面の横幅を求める  */
606:     max_len = 0;
607:     for(i=0;i<c_count;i++){ /* ファイル名の最大の長さを求める */
608:         tmp_len = strlen(c_files[i]);
609:         if(tmp_len>max_len) max_len = tmp_len;
610:     }
611:     max_len++;            /* 最大の長さより 1 だけ大きくする（空白用） */
612:     sprintf(fmt,"%%-%ds",max_len); /* printf 用のフォーマットを作る */
613:     width = width/max_len;  /* 何個ファイル名を表示したら改行するか */
614:     printf("¥n");
615:     for(i=0;i<c_count;i++){ /* 後は順番に表示するだけ */
616:         printf(fmt,c_files[i]);
617:         if((i%width)==(width-1)) printf("¥n");
618:     }
619:     if(i%width) printf("¥n"); /* 最後が改行で終わってないときは改行 */
620: }
621:
622: /*
623:  * カーソル位置より左にある単語で始まるファイル名があるなら残りを見つけて
624:  * 完全なものにする。また、それが第 1 カラムにある単語なら PATH の中から
625:  * 一致するコマンド名を探す。ファイルにしろ、コマンドにしろ、候補が複数
626:  * ある場合にはその一覧を表示する。
627: */
628: int complete(char buf[],int pos)
629: {
630:     int i;
631:     int dot;
632:     int part_len;
633: #ifdef    STANDARD
634:     char *name;
635:     struct {
636:         char   buf[150];
637:         char   atr;
638:         int    date;
639:         size_t len;
640:     } fbuf;
641: #else
642:     ctruct FILBUF fbuf;
643: #endif
644:
645: /* カーソルの下に何か文字があるときは何もしない */
646: /* つまり、行末か下が空白のときのみ処理する      */
647:     if((buf[pos]!=0)&&(buf[pos]!=' ')) return (pos);
648:     i = pos;
649:     dot = 0; /* 与えられた単語に . を含むか? */
650:     while((i>0)&&(buf[i-1]!=' ' ) /* とりあえずパス名の区切りまでの */
651:                 &&(buf[i-1]!='¥¥') /* 文字数を調べる */
652:                 &&(buf[i-1]!='/' ) /* その間に . があれば dot=1 に   */
653:                 &&(buf[i-1]!=':')){
654:         if(buf[i-1]=='.') dot = 1;
655:         i--;
656:     }
657:     part_len = pos-i;                 /* パス名以降の文字数   */
658:     if((i>0)||(buf[i-1]!=' '))        /* 単語の先頭を見つける */
659:         while((i>0)&&(buf[i-1]!=' ')) i--; /* 空白か左端まで */
660:     if((pos-i)>=25)                   /* 単語が長すぎると     */
661:         return (pos);                 /* 何もしない           */
662:     strncpy(word,&buf[i],pos-i);      /* 単語をコピーしてくる */
663:     word[pos-i] = 0;                  /* 文字列を終わらせる   */
664: #if 1
665: /* この #if から #endif はコマンドの補完を行う処理 */
666:     if(i==0){ /* 第 1 カラムからの単語 */
667:         c_count = cmd_complete(word,c_files); /* コマンドの補完 */
668:         if(c_count>0){     /* 候補がある場合 */
669:             if(c_count==1) /* 候補がひとつだけなら挿入して終わり */
670:                 return (f_insert(buf,pos,part_len,c_files[0]));
671:             if(f_common()){/* 共通な文字列があるならそこまでを挿入 */
672:                 f_insert(buf,pos,part_len,word_cmn);
673:                 f_print(); /* その後、一覧を表示して終わり */
674:                 return (pos-part_len+strlen(word_cmn));
675:             }
676:             f_print();/* 共通な文字列がなければ一覧を表示して終わり */
677:             return (pos);
678:         }
679:         /* 第 1 カラムからの単語でも候補がなければファイル名の補完へ */
680:     }
681: #endif
682:     if(dot) /* . がある場合とない場合でワイルドカードの付け方が違う */
683:         strcat(word,"*");    /* "単語"+"＊" */
684:     else
685:         strcat(word,"*.*");  /* "単語"+"＊.＊" */
686:     c_count = 0;
687: #ifdef STANDARD
688:     if((name=files(fbuf.buf,&fbuf.atr,&fbuf.date,&fbuf.len,word,0x3f))
689:        !=NULL){                                  /* 一致するファイル? */
690:         strcpy(c_files[c_count++],name);         /* あったらコピー */
691:         while((name=nfiles(fbuf.buf,&fbuf.atr,&fbuf.date,&fbuf.len))
692:            !=NULL)                               /* 残りがあるだけ */
693:             strcpy(c_files[c_count++],name);     /* それもコピー   */
694:     }
695: #else
696:     if(FILES(&fbuf,(unsigned char*)word,0x3f)>=0){ /* 一致するファイル? */
697:         strcpy(c_files[c_count++],fbuf.name);      /* あったらコピー */
698:         while(NFILES(&fbuf)>=0)                    /* 残りがあるだけ */
699:             strcpy(c_files[c_count++],fbuf.name);  /* それもコピー   */
700:     }
701: #endif
702:     if(c_count>0){ /* 候補がある場合 */
703:         if(c_count==1){                  /* 候補がひとつだけで */
704:             if(fbuf.atr & 0x10)          /* ディレクトリの場合 */
705:                 strcat(c_files[0],"¥¥"); /* 名前の後ろに ¥ を付ける */
706:             return (f_insert(buf,pos,part_len,c_files[0])); /* 終了 */
707:         }
708:         if(f_common()){  /* 候補の先頭に共通の文字列が含まれる場合 */
709:             f_insert(buf,pos,part_len,word_cmn); /* それを挿入して */
710:             f_print();                          /* 一覧を表示     */
711:             return (pos-part_len+strlen(word_cmn)); /* 終了 */
712:         }
713:         f_print(); /* 複数の候補がある場合は一覧を表示 */
714:     }
715:     return (pos);
716: }
717:
718: /*******************************************************
719:   コマンド名補完処理の実体
720: ********************************************************/
721: static char nam_head[90];
722: static int  dot_end;
723:
724: static int name_check(char name[])
725: {
726:     /* ドライブ名、パス名、あるいはドットがないことをチェック */
727:     /* ドットがファイルの最後ならＯＫ */
728:     char *p=name;
729:     char *q=nam_head;
730:
731:     dot_end = 0;
732:     if(*p=='.') return (0); /* いきなりドットは困る */
733:     while(*p){
734:         if((*p==':')||(*p=='/')||(*p=='¥¥')) return (0);
735:         if(*p=='.'){
736:             if(*(p+1)==0)
737:                 dot_end = 1;
738:             else
739:                 return (0);
740:         }
741:         *q++ = *p++;
742:     }
743:     *q = 0;
744:     return (1);
745: }
746:
747: static int cmd_find(char cfiles[][25],int count,char p_name[])
748: {
749:     char tmp_name[90];
750: #ifdef    STANDARD
751:     char   *name;
752:     char   buf[150];
753:     char   atr;
754:     int    date;
755:     size_t len;
756: #else
757:     struct FILBUF fbuf;
758: #endif
759:     char *ext0[]={"*.r","*.z","*.x","*.bat"};
```

▶X68000のカバーを1枚外しただけで，より親しみを感じるようになりました。早く，改造できる知識を身につけたい。　半田　文男(18)香川県

```
760:     char *ext1[]={"r","*z","x","bat"};
761:     char **ext;
762:     int  i;
763:
764:     if(dot_end) ext = ext1;
765:     else        ext = ext0;
766:     for(i=0;i<4;i++){
767:         strcpy(tmp_name,p_name);
768:         strcat(tmp_name,nam_head);
769:         strcat(tmp_name,ext[i]);
770: #ifdef STANDARD
771:         if((name=files(buf,&atr,&date,&len,tmp_name,0x20))!=NULL){
772:             /* 普通のファイルのみを検索 */
773:             strcpy(cfiles[count++],name);
774:             while((name=nfiles(buf,&atr,&date,&len))!=NULL)
775:                 strcpy(cfiles[count++],name);
776:         }
777: #else
778:         if(FILES(&fbuf,(unsigned char*)tmp_name,0x20)>=0){
779:             /* 普通のファイルのみを検索 */
780:             strcpy(cfiles[count++],fbuf.name);
781:             while(NFILES(&fbuf)>=0)
782:                 strcpy(cfiles[count++],fbuf.name);
783:         }
784: #endif
785:     }
786:     return (count);
787: }
788:
789: int cmd_complete(char head[],char cfiles[][25])
790: {
791:     char *path_s,*path_e;
792:     char p_name[90];
793:     int  f_count=0;
794:
795:     if(name_check(head)==0) /* 補完できないファイル名なら何もしない */
796:         return (0);
797:     if((path_s=(char*)getenv("PATH"))==NULL)
798:         path_s=(char*)getenv("path"); /* PATH がなければ path をみる */
799:     if(path_s==NULL)                  /* 環境変数 PATH がない */
800:         return (0);
801:     path_e = path_s;
802:     while(*path_e){
803:         if(*path_e==';'){
804:             strncpy(p_name,path_s,path_e-path_s);
805:             p_name[path_e-path_s] = 0;
806:             if(p_name[path_e-path_s-1]!='¥¥') /* PATH の区切りをつける */
807:                 strcat(p_name,"¥¥");
808:             f_count = cmd_find(cfiles,f_count,p_name);
809:             path_s = path_e+1;
810:         }
811:         path_e++;
812:     }
813:     if(path_s==path_e){ /* こんなことはないと思うけど */
814:         return (f_count);
815:     }
816:     strncpy(p_name,path_s,path_e-path_s);
817:     p_name[path_e-path_s] = 0;
818:     if(p_name[path_e-path_s-1]!='¥¥') /* PATH の区切りをつける */
819:         strcat(p_name,"¥¥");
820:     f_count = cmd_find(cfiles,f_count,p_name);
821:     return (f_count);
822: }
823:
824: /*************************************************************
825:      標準入出力リダイレクト処理の実体
826: *************************************************************/
827: /*
828:  *     ここでリダイレクトは
829:  *         fopen -> dup2
830:  *     という手順で行っている
831:  *          freopen
832:  *      ではコプロセスの標準入出力をリダイレクトできなかった
833: */
834:
835: /* 標準入出力リダイレクト用のファイル名 */
836: static char std_in [150];
837: static int  in_mode;
838: static int  in_dir;
839: static char std_out[150];
840: static int  out_mode;
841: static int  out_dir;
842:
843: /* 標準入出力記憶用のファイルハンドル */
844: static int  fhi;
845: static int  fho;
846:
847: /* 標準入出力切り替え用のファイルポインタ */
848: static FILE *fpi;
849: static FILE *fpo;
850:
851: /*
852:  * ファイル名を取りだし、cmd_argv の中から削除する
853: */
854: /*
855:                 after
856:                   v
857:     [?][?][<][F][?][?][?][0]
858:             ^               ^
859:          dir_pos       cmd_argc
860:
861:                 after
862:                   v
863:     [?][?][?][?][?][0][?][0]
864:             ^               ^
865:          dir_pos       cmd_argc
866:
867: */
868: static int assign_in(char *file,int dir_pos,int after)
869: {
870:     int i;
871:
872:     if(std_in[0]!=0){
873:         fprintf(stderr,
874:             "標準入力への割り付けが重複しています。¥n");
875:         return (-1);
876:     }
877:     strcpy(std_in,file);
878:     for(i=0;i<=(cmd_argc-after);i++)
879:         cmd_argv[dir_pos+i] = cmd_argv[after+i];
880:         cmd_argc -= (after-dir_pos);
881:     return (1);
882: }
883:
884: static int assign_out(char *file,int dir_pos,int after)
885: {
886:     int i;
887:
888:     if(std_out[0]!=0){
889:         fprintf(stderr,
890:             "標準出力への割り付けが重複しています。¥n");
891:         return (-1);
892:     }
893:     strcpy(std_out,file);
894:     for(i=0;i<=(cmd_argc-after);i++)
895:         cmd_argv[dir_pos+i] = cmd_argv[after+i];
896:         cmd_argc -= (after-dir_pos);
897:     return (1);
898: }
899: /*
900:  * リダイレクトするファイル名を取り出す
901:  * 簡単のため、> や < が 1 個以上続くとアペンドとみなしている
902:  * << は通常と意味が違う。単に < と同じにしている
903: */
904: int redirect(void)
905: {
906:     int i;
907:
908:     std_in[0]  = 0;
909:     in_mode    = 0;
910:     in_dir     = 0;
911:     std_out[0] = 0;
912:     out_mode   = 0;
913:     out_dir    = 0;
914:
915:     for(i=0;i<cmd_argc;i++){
916:         if(cmd_argv[i][0]=='<'){
917:             if(cmd_argv[i][1]==0){
918:                 if((i+1)>=cmd_argc){
919:                     fprintf(stderr,
920:                         "標準入力に割り付けるファイル名がありません。¥n");
921:                     return (-1);
922:                 }
923:                 if(assign_in(cmd_argv[i+1],i,i+2)<0)
924:                     return (-1);
925:                 i--;                  /* 同じ位置の要素から解析をやり直す */
926:             }
927:             else if(cmd_argv[i][1]=='<'){ /* << 直接入力 */
928:                 in_mode = 1;
929:                 out_mode = 1;               /* アペンドモードを指定    */
930:                 cmd_argv[i] = &cmd_argv[i][1]; /* < を 1 個けずってやり直し */
931:                 i--;
932:             }
933:             else{ /* < の直後にファイル名が続く場合 */
934:                 if(assign_in(cmd_argv[i]+1,i,i+1)<0)
935:                     return (-1);
936:                 i--;
937:             }
938:             continue;
939:         }
940:         if(cmd_argv[i][0]=='>'){
941:             if(cmd_argv[i][1]==0){
942:                 if((i+1)>=cmd_argc){
943:                     fprintf(stderr,
944:                         "標準出力に割り付けるファイル名がありません。¥n");
945:                     return (-1);
946:                 }
947:                 if(assign_out(cmd_argv[i+1],i,i+2)<0)
948:                     return (-1);
949:                 i--;
950:             }
951:             else if(cmd_argv[i][1]=='>'){ /* >> アペンド */
952:                 out_mode = 1;               /* アペンドモードを指定    */
953:                 cmd_argv[i] = &cmd_argv[i][1]; /* > を 1 個けずってやり直し */
954:                 i--;
955:             }
956:             else{ /* > の直後にファイル名が続く場合 */
957:                 if(assign_out(cmd_argv[i]+1,i,i+1)<0)
958:                     return (-1);
959:                 i--;
960:             }
961:             continue;
962:         }
963:     }
964:     if(std_in[0]!=0){ /* 標準入力のリダイレクトあり */
965:         fhi = dup(fileno(stdin));
966:         fpi = fopen(std_in,"r");
967:         if(fpi==NULL){
968:             fprintf(stderr,
969:                 "%s: 標準入力に割り付けるファイルがオープンできません。¥n",
970:                 std_in);
971:             return (-1);
972:         }
973:         if(dup2(fileno(fpi),fileno(stdin))<0){
974:             fprintf(stderr,
975:                 "%s: ファイルが標準入力に割り付けできません。¥n",
976:                 std_in);
977:             return (-1);
978:         }
979:         in_dir = 1;  /* ファイルがリダイレクトされた */
980:     }
981:     if(std_out[0]!=0){/* 標準出力のリダイレクトあり */
982:         fho = dup(fileno(stdout));
983:         if(out_mode)
984:             fpo = fopen(std_out,"a");
985:         else
986:             fpo = fopen(std_out,"w");
987:         if(fpo==NULL){
988:             fprintf(stderr,
989:                 "%s: 標準出力に割り付けるファイルがオープンできません。¥n",
990:                 std_out);
991:             if(in_dir){ /* 標準入力のリダイレクトを元に戻す */
992:                 fclose(fpi);
993:                 dup2(fhi,fileno(stdin));
994:                 in_dir = 0;
995:             }
996:             return (-1);
997:         }
998:         if(out_mode)
999:             fseek(fpo,0,SEEK_END);      /* ファイルの最後にシーク */
1000:         if(dup2(fileno(fpo),fileno(stdout))<0){
1001:             fprintf(stderr,
1002:                 "%s: ファイルが標準出力に割り付けできません。¥n",
1003:                 std_out);
1004:             return (-1);
1005:         }
1006:         out_dir = 1; /* ファイルがリダイレクトされた */
1007:     }
1008:     return (1);
1009: }
1010:
1011: /*
1012:  * 標準入出力のリダイレクトを回復
1013: */
1014: void reredirect(void)
1015: {
1016:     if(in_dir){
1017:         fclose(fpi);
1018:         dup2(fhi,fileno(stdin));
1019:     }
1020:     if(out_dir){
1021:         fclose(fpo);
1022:         dup2(fho,fileno(stdout));
1023:     }
1024:     std_in[0]  = 0;
1025:     std_out[0] = 0;
1026:     in_dir     = 0;
1027:     out_dir    = 0;
1028: }
```

▶うーんローテク。私は，こういった小手先の改造がだ～い好き。
弘田　永幸(25)神奈川県

CREATIVE COMPUTER MUSIC

Creative Computer Music入門(23)

それでも採譜ができません

1992年8月号の本連載で採譜のやり方を解説してから，ちょうど1年。そのあいだ，さまざまな音楽理論を勉強してきました。しかし，理論はわかってもなかなか実践に直結できないこともありますよね。今月は「実践」のひとつとして，もう一度，採譜について考えてみます。

Taki Yasushi 瀧　康史

§好きな声がありますか？

皆さんには，好きな声ってありますか？

いきなりこんなことをいわれても，なんのことかわからないですよね。それでは，ヴォーカリストの「声」そのものが聴きたくて音楽を聴くことはありますか？　さして歌がうまいわけではないけれど，この人の声が好き，というのは絶対にありますよね。

かくいう私にもそういうのがあります。以前にCDの紹介で話にのぼったバーシア（女性ヴォーカリスト）は声が色っぽくて好きですし，ほかにも，マルティカ（同じく女性ヴォーカリスト）なんかは声にハリがあってとっても気にいっています。あと，女性でいえば，チャカ・カーンのヴォリュームのある声やティファニーなんかのハスキーヴォイスなどなど。男性ならば，シナトラなんか，男性に対して使っていいのか迷っちゃう言い方ですけど艶っぽくて好きだし，そうやって挙げたらきりがありません。

こういった人たちは「歌」でご飯を食べているような人ですから，声だけじゃなく，歌い方などに魅力があるのでしょう。ひょっとしたら，私も「声が好き」なのではなくて「歌がうまいから好き」なのかもしれません。

ですが，決して歌がうまいといえないような人の曲を聴くこともしばしばあります。友人から，「どうしてそんなにヘタクソで，顔もよく知らないアイドルの歌なんて聴くの」といわれたことがありますが，その声が好きなんだからしかたがないのです。まあ，名前は伏せておきますが，その人の声を聴くと心が和むというのはあるんですよね。

以前，別の友人と，「やっぱり一度はアイドルミュージックに手を出すよね」という話をしたことがあります。自分の信じる道をつき進み，ロックばっかり，クラシックばっかり，ジャズばっかりというような人は確かにいるかもしれませんが。その友人の体験談では，浪人時代に某女性アイドルの「なんにも悩みのなさそうな歌声」を聴いて心が和んだことがあったそうなのです。もちろん歌ってる本人に悩みがないなんて思えませんから，それはそれ，彼女の才能なのでしょう。

私はテレビを持っていないので，よくはわかりませんが，さらに別の友人の話だと，その人はある声優の声が好きで，よくアニメを観るのだそうです。なんか本筋から外れているような気もしますが，そういうこともあるのでしょう。そういえば，声がよく耳についてしまう人というのもいますからねぇ。

さて。

皆さんには「好きな声」がありますか？

§うまくいかない!?

ちょうど1年前に，採譜について，その方法を教えてほしいというリクエストがありました。そのときに，私の立場からいえることは一応すべて述べたつもりなのですが，そのあとも，パソコン通信の場や音楽仲間との話のなかで，それでも採譜はうまくいかないと嘆いている人をみかけます。

その人の実力不足なのか，私の連載を読んでないか，私の説明が足りなかったのか，そのあたりはわかりませんが，今回もう一度，採譜のコツについてやることにしました。

こういうとしかたなくやっているようですが，それだけではありません。実は前回に書き残したことがいくつかあり，それについていつ補足しようかと迷っていたのです。また，前回の話のなかで，音のメロディぐらいは自分で採れ，というようなことを書きましたが，それだけでは，結果的にメロディが採れなかった人には何の解決にもなりませんよね。そういうことで，もう一度と考えたのです。

私自身，音を採るという作業は，楽しいながらなかなかつらいものと考えています。何でもそうでしょうが，わかればうれしい反面，同じところで何度もつまずいてしまうと，どうにも嫌になってしまうからです。

そこで弱気な人は，自分は採譜には向いてないかもしれないと考えてしまうのでしょうね，きっと。でも，私もこのように音楽理論などを解説する身でありながら，うまくいかないときはうまくいきません。

そのようなときは，「ゲンをかつぐ」のに近いようなことを何度もやったり，知り合いに尋ねたり，音楽のことを何も知らない人に聞いてみたりと，ありとあらゆることをやってみます。

調子が悪いときなどは，自分自身では絶大なほど自信を持っているにもかかわらず，結果としてはうまくいってない場合があります。人に指摘されてはじめてそれに気がつくなんてこともあります。たとえ絶対音感を持っている人でも，いつでも同じ周波数でオクターブを歌えるわけはないのですからね。

そこで，今回はこうした，役に立つか立たないか，ジンクスやらなにやらを含めて，音が採れるまでの過程をいろんな面から追ってみたいと思います。

前回を読んでいない人でもわかるようにするために，重複して書く部分もありますが，1992年8月号を持っている人は，一度それに目を通してからこちらを読むと，私がどの点について書き逃したかがわかって，おいしさは2倍です。逆に，この2つの号を読んでもまだ採譜ができない人がいたらお便りください(質問は，PC-VANのSIG，X1CLUB音楽室で随時受け付けています。全国ネットなので，北海道でも沖縄でもたぶん安心。ちなみに，ここではZ-MUSICのサポートもやっています)。

では始めましょう。

§ターゲットを選ぶ

説明はいきなりメロディから入りますが，採譜するのは，別にメロディからでなくてもかまいません。前回は，拍を採って，メロディを採って，ベースを採って……うんぬんと書いたような気がしますが，とりあえずそれは忘れてください。あれはひとつのパターンであって，自分の好きな順序で採ってかまいません。わかりやすいものから順に採ればよいのです。

さて，まずはターゲットとなる曲を選びます。なになに？　CD「超兄貴－兄貴のすべて－」の中に入っている「あこがれのマッチョダンディー」？　おーけーおーけー。何でもおーけー。ここで，音数が少ないほうが簡単かな，なんて思っちゃいけません（この前といっていることが違いますけど)。意外にも，音数が少ないもののほうが難しい場合もあるし，バリバリのロックでも曲によっては楽なものもあります。ですから，あまり細かいことは気にしないで，適当に，できるだけ自分の「好きな」曲を選んでください。

なぜって。ターゲットとなる曲が好きな曲ならば，それだけ頻繁に聴いているはずだからです。

音楽を聴いて，最初から順に採譜しようとすると，よほどの天才でもない限り，無駄な労力を浪費すると思います。

そこでまず，ポータブルのCDやテープなどを持ち歩き，よく聴いてみてください。そしてその曲を「好きになる」こと。CDならばループ機能があって，同じ曲が何度も聴けるから，とにかく曲を「すみずみまで」聴く。すみずみまで聴いたら，きっと普通ならば聴き流してしまうようなパートについてもわかってきます。たとえばロックなら，ベースがここでおいしいことをしてるとか，ギターがここでおいしいことをしているとか，そういったことを聴き逃さないでほしいのです。

そうです，この作業を繰り返すのはそれほど難しいことではないはずです。好きな曲なんですから。そして，じっくり聴き込めば，それまでは漠然としていたその曲のすみずみまでの魅力が，はっきりとわかってくるはずです。そうして何度も聴いているうちに，どのパートもだいたい口ずさめるかなといった感じがしてくるでしょう。ほら，誰しもそういう曲が1つぐらいはあるでしょう？　好きで，何度も聴いて，聴いて，聴いて。その結果，どこからどの楽器が入ってくるとかそういうところがわかるまで何度も聴き込んでる曲が。すでにそういったものがあるのなら，その曲をターゲットにするのが，採譜の勉強ではいちばんの早道でしょう。それにそういう曲はきっと採譜も楽しいだろうし。

前回のターゲットの選び方よりも楽でしょう？

§歌いながら音を採る

音採りをするんだったら，歌えなくちゃだめ。

でも，ここでいう歌い方というのは，別に「愛のこもった」歌い方などを求めているわけではありません。外れていないであろう音程でメロディやそのほかのパートを口ずさめるだけでよいのです。これは，ターゲット曲を選んだ段階ですでにだいたいできていると思います。歌うことができたら，自分の歌っている音がいったい何の音なのか認識する必要があります。つまり，自分がいま歌っている曲を紙の上に残さねばならないからです。それを「採譜」といいます。自分が歌っている歌ならば採れるんじゃないかと思いませんか？

さてこの作業は，最初からメロディの流れる順番どおりにやろうと思い込んではいけません。今回は，採る順序を決めてあるようにみえますが，これは説明するうえでの便宜的なもので，実際の作業順序はもっと入り組んだものになります。最初の段階では，この小節ではリズムしかわからないとか，この小節ではメロディしかわからないといった状態のままでよいのです。そして，わかるところから順に楽譜ノートに書き込みます。こちらのほうがずっと効率がよいですから。簡単なところをやっているうちに，その曲の特徴が見えてきて，最初はわからなかった部分が，そのうちにわかってくるなんてことは日常茶飯事だからです。

それから，メロディは音の連続だということを忘れて

はいけません。いくら絶対音感のある人でも，すぐ1拍前にEがあるCと，独立したCとではどちらがわかりやすいか考えてみてください。想像に難くないですよね。だから，もしもどうしても採れない小節があれば，その前の音や後ろの音から相対的に考えてみるとわかりやすいでしょう。数学的帰納法がこの場合でも成り立つのならば，これで，わからない音というのはなくなるといえます(でも，我々は人間である，と)。わからない音が完全になくなるかどうかは別として，ほとんどの部分はこれでわかるようになるでしょう。

前回では，ループ機能を使ってわからないところを何度も再生してみたらよい，というようにいいましたが，これは訂正します。これをやると，「はまる」場合があるからです。実は，私もつい最近はまってしまいました。同じところだけ100回も200回も聴いたら，正しく歌えているはずなのに，その部分の音が一向に採れなくなってしまったのです。音楽は音の連続ですから，やっぱり連続性が重要なんですね。通して聴いてみたほうが楽でした。もっとも，人あるいは曲によっては，同じところだけ繰り返して聴くほうがよい場合もあるかもしれませんけれど。

さて，歌うことができたなら，すなわちそれは正しく音を採れている証拠です。メロディの採譜を2つのプロセスに分けるとするならば，まず音の高さを採り，ついで音の長さを採るということになるでしょう。音の高さを採るということは，歌うことですでにマスターできています。これを楽譜上に置くということは，つまり自分の歌っている音程がなんの音なのかを知ることですから，吹奏楽器では不可能ですけど，鍵盤楽器などなら歌いながら鍵盤を弾くことで音を採ることができます。ギターなどでもかまいません。残念ながらこのことは努力が必要な人がいるかもしれませんが，小・中学校の音楽課程のなかで，先生のピアノに合わせて歌を歌うという学習はしているはずですから，それができていたのならば，取っかかりに多少困難があったにしても，次第に慣れるでしょう。

それよりも，問題は音の長さを知ることです。4分音符や8分音符などだけで記述された曲ならば簡単なのですが，曲のなかに16分音符や2分音符などいろいろな長さの音が効果的に入り乱れていると，採譜はなかなか簡単にはいかなくなります。特に難しいのは16分音符と8分音符の組み合わせが続くような節で，そんなものが最初にきてしまうと必ず苦労します。

そういう曲は採譜しにくい曲であることに変わりはありませんが，要はちょっとしたコツで，それなりに作業を楽にすることは可能です。これは同時に，リズムをつかむことにつながりますが，まず最低，拍かそれに近いものをしっかりと得なければなりません。

たとえば，次の楽譜を見てください。

この短い楽譜のメロディを採るとして，8分音符を基準に考えるとどうでしょうか。

8分音符でリズムを採り，4分音符はその2つ分と考えれば簡単です。付点4分音符ならば3つ分と考えれば，これもまた容易に採れます。

ところが，ここで16分音符が出てくると，8分音符を基準にしたリズムでは一筋縄ではいかなくなります。そこで，その部分だけは基準を16分音符として考えてみます。すると，付点8分音符も簡単に採れます。

つまり，音の長さについては，いちばん短い音を基準にして，それぞれの音については基準音のいくつ分なのか，というように考えればよいのです。しかし，最初から短い音を基準音として定めると，逆に2分音符などの長い音がきたときには大変ですから，それについては採譜しながら臨機応変に考えてください。

§リズムを採る

リズムを採ることと音を採ることには，密接な関係があります。リズムを採るというのは，「ドラムスを採る」などということではなく，曲の拍子などを確実に採ることです。

したがって，一般にいうリズム担当楽器の楽譜を採るときに最も注意しなければならないことは，最短音符の長さです。

では，具体的にはどのように考えたらよいのでしょうか。曲の最初にたとえば16分音符があって，それがその曲のなかでの最短音符なら話は簡単なのですが，もちろん，そういう場合ばかりではありませんよね。以下のようなリズムパターンのときにも16分音符が最短音符だということを理解してください。

要するにこの例は，付点8分音符が2つあり，ただの8分音符につながるといった簡単なものですが，この場合の最短音符（というと語弊があるかもしれませんが）は16分音符になります。

この場合，16分×3，16分×3，16分×2，……，というように採ります。

では，このようにして最短音符を決め，曲中でどのようにしてリズムが連なるかを考えながら，もう一度ターゲットとなる曲を聴いてみてください。そのあとで実際に，いまいったことを踏まえてリズムの楽譜を採れば，比較的簡単に採譜できるでしょう。

§和音の変異

理屈では，以上に述べた2つのことを考慮すれば，曲の楽譜はすでに採れているはずです。が，実際にはなかなかうまくいかないことも多いでしょう。たとえば，音がたくさん重なりすぎて，いったいどんな音が鳴っているのかわからないとか，いろいろあるはずです。

私たちはいままで和声学などを中心に勉強してきました。そこでその和声学を利用して，採譜が楽にできるか考えてみましょう。

和声を採るにはまず，その曲のスケールを知ることが必要となります。

スケールを知るための最も単純な方法は，ベースパターンから探ることです。ベースパターンは簡単にいえば，最も低い音ですから，バスドラムなどの音を除いたいちばん低い音をそのまま採れば，それがベースパターンのヒントになります。

前にもいいましたが，最近のオーディオ機器はバスブーストなどが付いていてベースを目立たせることができるので，比較的採りやすいパートといえるでしょう。

もはやいうまでもありませんが，このベースパターンからベースノートは導き出されます。ベースパターンは，何よりもベースノートを確実に押さえなくてはならないからです。ベースノートとはコードのいちばん下の構成音です。つまりそれはコードの決定を意味しているため，たいへん重要な役割をもつことになります。

とりあえず，ベースパターンとベースノートの関係について，図をもって説明しましょう。

上の図のなかで，Aパートは楽器がそのまま演奏しているもの，つまりここではベースパターンといっているものです。対して，Bはこのベースパターンが押さえているベースノートを表します。

採譜の第一目的は楽譜を採ることですから，ここでの場合AすなわちベースパターンをJ採ることなのですが，スケールを知り，最終的に和声進行まで知るためには，Bのベースノートを理解することも必要です。

同時にこの2つが採れてしまえばよいのですが，実のところ，ベースパターンよりもむしろ，ベースノートのほうが採りやすいことがしばしばです。

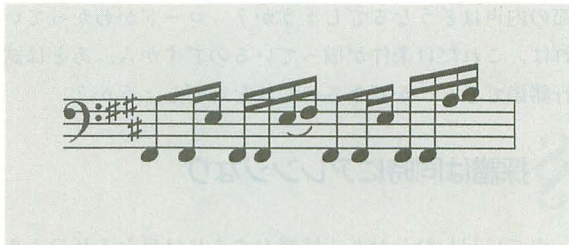
ここでベースノートと，ベースパターンの関係をまとめてみますと，

・ベースパターンの小節の最初の拍はベースノートである場合が多い

・小節のなかで最も多い音はベースノートの場合が多い

と，このようになります。

また，仮にベースパターンが動いたとしても，コードの構成音のなかを動くか，それらを経過音でつなぐことが多く，実際に上下にふられているベースパターンなどは，一見複雑にみえても，コードの構成音を往復しているだけだということが多いのです。

とりあえず次のベースパターンを見てください。

実は，これは某バンドの某曲の某一節なのですが，まぁこれだけではわからないでしょう。たった1小節ですし。これのベースノートはいわずもがな，F#です。強拍には必ず押さえられていて，かつ，いちばん出現確率が高い音ですから一目瞭然でしょう。

実際にこの楽譜どおりに音を鳴らすと，このベースパターンはなかなか散っているのですが，このとおり基本は押さえられています。しかも，よく見ると，実はこのベースパターンは，ベースノート以外の音は経過的に連なっていることがわかるでしょう。

さて，ベースノートがわかると，スケールがわかります。ベースノートでもスケールでも，どちらから導き出してもかまいませんが，ベースノートの特性を利用することによりスケールを探ることができるというのも事実です。実際，この例の1小節は曲の冒頭で，曲のスケールはF#です。

また，スケールというものがトニックと等しいことは，これまで学習してきた人には容易にわかるでしょう。そしてベースノートですが，大半の場合，コードの根音が「多い」という事実があります。転回形がたまにありますが，これはたいていは偶成和音によるものです。

また，これも以前いいましたが，多くの場合，曲はト

ニックから始まります。つまりたいていは，最初のベースノートでその曲のスケールが把握できることになります。

たとえトニックから始まらない曲であっても，トニックは，何かあれば必ず戻るコードですから，曲の節々を追っていけば必ずわかるはずです。

問題は内声です。

いままでの要領でベースノートは採れました。トップノートはメロディの場合が大半だからうまくいったということにして，わかることをすべて楽譜に書き込んでしまうと，あとはたいてい内声部，しかもメロディアスでない部分が残っていると思います。こういった部分を採る決定的な手段はないのですが，それでも楽をする手段を考えてみましょう。

まずは，全体のコード進行をだいたい把握できていることとして話を進めます。

前後があるのならば，前後を四声体に分離します。前回の採譜のときには，まだ四声体を学習する前だったのでこのことについては省いてあったのですが，今回は大丈夫でしょう。前後の四声体がわかった場合，不明な小節の内声はどうなるでしょうか？　コードがわかっていれば，これだけ条件が揃っているのですから，あとは試行錯誤でどうにかできるのではないでしょうか？

§採譜は同時にアレンジなり

すでに記したとおり，採譜をするには単なる耳のよさだけではなく，アレンジャーとしての耳のよさも問われてきます。

なにしろ，外声に隠されている内声は，ほとんど「カン」で採っていますから。このときに必要な「採譜」のときの耳というのは，すなわち原曲が行っているアレンジを発見することにほかなりません。

たしかに，まったくの真似に近くなるまで試行錯誤して内声を「決定」するのは骨が折れますが，それなりの勉強になります。私などはついついすぐ逃げてしまい，正確な採譜というよりも自分なりのアレンジにしてしまうのですが……。

それでも，採譜という勉強を通してアレンジの技術を身につけたということになるのですから，これもまた，瓢箪(ひょうたん)から駒かもしれませんね。

内声の決定は作曲と同じぐらい面倒くさいものですが，上記のテクニックと，和声的理論，そして，あなたの感性でがんばってください。

それから，いい忘れていましたが，この内声を決定するときには，内声同士のアンサンブル，果ては外声とのアンサンブルに注意する必要があります。したがって，楽譜を書いたらさあ終わり，という環境ではなく，楽譜を書いたら即演奏できる環境にしておいたほうが遥かに楽です。これにはいろいろな方法がありますが，私は最近，「Musicstudio PRO-68K」と手書き楽譜の組み合わせが多いですね。「MUSIC PRO-68K」のようなソフトがもっと高機能ならよいのですが……。覚えてしまうと，ZMSファイル＋手書き楽譜の組み合わせも悪くはないですよ。

§まとめ

前回，つまり1992年８月号に掲載した「効果的な採譜のやり方」の追補版のようなノリで始めたのですが，その号を読んでいない人にも理解できるように書いたつもりです。いかがでした？

すでに，連載は理論の学習を通り過ぎ，応用のレベルに達しているので，あとは細かい理論は省略することにしています。いつまでもうだうだとコードについて書くと，すでに理解している人にはうっとうしいだろうと思ったからです。それでも私自身，いま現在でいえることをまとめてみたつもりなのですが，場合によってはわかりにくいかもしれません。もうちょっと基礎的なことにも触れたほうがよかったのかもしれません。

いずれにしても，採譜に関するテクニックというのは以上のようなことでほとんどすべてではないでしょうか。あとは本当に「愛」と「根性」の世界です。「わかってるのにわからない!?」なんていう台詞を何度も吐くことになるでしょう。私もそういいながら，採譜したりしましたからね。

ところで，私事ですが，最近PC-9801NLという薄形ノートパソコンを買いました。ちっちゃくてなかなか快適なのですが，これが脆くてひと月しないうちに液晶が割れてしまって，もうさんざん。みんなからは使い方が悪いんだといわれましたが，小物に愛情をそそぐほどいまは余裕がないのでした。あ～あ。このノートには今月号の原稿をすっかり消されてしまいましたからね。なんかのバチが当たったのかなあ（それでも修理費を払うのは自分……)。

そうそう，冒頭の「好きな声」ですが，私はアリッサ・ミラノの声が大好きです。まぁ，声が大好きなのであって，人物が好きかどうかは別ですが。ちなみに日本人ならば，工藤静香がいます。最近，彼女は歌がうまくなって「声の」ファンとしては幸せです。このままボリュームある声になってほしいなぁ。なぜだか知らないけど，この人の声はよく耳に残るんだよなぁ。

さて，来月，もしくはさ来月以降については，やりたいことは山ほどあるのですが，どうなるかはまだ決めていません。リクエストのあった，曲の切り出し方や，ジャンルを超えたリズムパターン，ベースパターンなどなど……いっぱいありますが，どうしましょうか。

では，来月。

THE SENTINEL

〈対応機種一覧〉●MZ-80 K/C/700/1500●MZ-80 B/2000●MZ-2500/2861●X1●X1 turbo/Z●PC-8001/8801/88●SMC-777/C●PASOPIA/5●PASOPIA/7●FM-7/77/AV●MSX/2/2+/turbo R●PC-286/386/486/9801/98/9821●X68000/X68030
掲載されたプログラムの利用には各機種用のS-OS"SWORD"システムが必要です。

第134部 MACINTO-C再掲載

●ソースリストの配布

7月号で予告していたMSX用S-OS"SWORD"のソースリストは，やはりページの都合により掲載が不可能となってしまいました。そこで，今回は特例として希望者にのみ，ディスクでソースリストの配布を行います。メディアは3.5インチ2DDのみで，希望者は官製ハガキに住所，氏名，年齢，電話番号，そして「MSX用S-OS "SWORD" ソースリスト希望」と明記のうえ，下記の宛先まで応募してください。

<宛先>
Oh!X編集部
「MSX用S-OS "SWORD"
ソースリスト配布」係

なお，申込期限は8月17日（消印有効）となります。発送は先着順に行う予定ですが，希望者多数の場合は発送が遅れる場合がありますので，あらかじめご了承ください。

また，これはあくまでもソースリストのみですので，オブジェクトは含まれません。お間違えのないように。

●THE SENTINELの明日は……

S-OSも今回で第134部，全機種共通システムが開始されてから，実に8年以上の月日が経過しました。

そもそも，全機種共通システムは，乱立していた8ビット機のどの機種でも，動くようなシステムを作ることが目的でした。当時，各メーカーが，それぞれのマシンを独自の規格で作っており，ソフトという資源がその単一機種でしか有効ではなかったのです（これは現在も変わってませんね）。そこで，機種は違っても同じCPUを搭載しているのならば，共通のOSを作ることでどの機種でもソフト資源を共有できるのではないか，そう考えたのです。

そして，S-OSが生まれました。機能的には，マシン語モニタと呼べる程度のものでしたが，その上には多くのアプリケーションが誕生しています。

これは，当時のパワーユーザーたちが，この全機種共通システムに賛同した成果といえるものです。つまり，THE SENTINELの業績は，すべて読者によって創り上げられたといっても過言ではありません。

その後は，皆さんご承知のとおりS-OSと名づけられたこの小さなOSは，止まることを知らないかのように次々とその勢力を伸ばしていきました。同じZ80ならいざ知らず，異なるCPUでもZ80をエミュレートしたり，専用のボードで対応したり……さまざまなアプローチが行われたのです。

しかしながら，現状では8ビット機の減少により，その存在意義を疑う声もあります。確かに，時代にそぐわなくなったから，という適当な理由づけを行って切り捨てることは簡単です。

ただ，その切り捨てを行ったとき，その切り捨てられた精神がどこへ受け継がれていくか，それがはっきりしないかぎり思い切った行動はとれません。なにかに受け継がれなければ，いままで築き上げてきたものが無駄になってしまうのです。

●これからのTHE SENTINEL

さて，THE SENTINELの軌跡はこれぐらいにして，少し前向きにこれからのS-OSについて考えていきましょう。

まず，THE SENTINELですが，投稿数の激減により状況はあまりよくありません。とだけいっておきます。とりあえず，来月はSLANGの再々掲載を予定しています。

そして，なんとかいままでのS-OSシステムアプリケーションをなんらかの形でまとめ，配布することを実施するしもしれません。以前から声高に叫ばれているこの企画を具体的に検討します。もしかしたら，この計画のために専用スタッフを募集するかもしれません。

東京近郊でS-OSに詳しい方，やる気のある方，S-OSサークルの方々，とりあえず愛読者アンケートハガキにその旨を書いてお送りください。

両方とも正式な決定は来月以降，このTHE SENTINELのコーナーで発表したいと思います。

読者の皆さんの協力がなくては成り立たないこの企画，あなたも積極的に参加してみませんか。

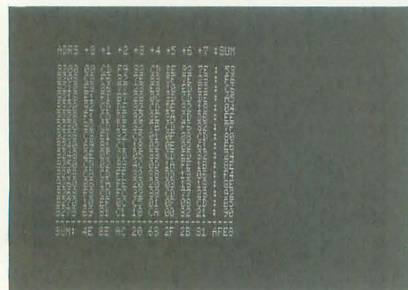
1993■インデックス

全機種共通

S-OS"SWORD"要

MACINTO-C

再掲載

S-OS "SWORD"の世界では，基本的にすべてのアプリケーションを自分で入力する必要があります。今月は，その基本入力ツールである「MACINTO-C」の再掲載をします。

MSX用S-OS"SWORD"が発表され，新しくS-OSの世界を体験する方もいると思います。そこで，今月はマシン語入力ツールである「MACINTO-C」を再掲載します。

今回は，3000_H版と$B000_H$版の2つのリストを掲載しました。機能的にはまったく同じものですので，必要に応じてどちらかのリストを打ち込んでください。短いリストなので一気に両方打ち込んでおくのもいいでしょう。アプリケーションによってはアドレスが重なることもあるので，両方入力しておけばあとあと便利です。

入力方法は，7月号で掲載された「MACMSX.BAS」を使うか，モニタから直接打ち込んでもらってもかまいません。とりあえず，入力が終了したらデバイスにセーブし，実行してください。起動アドレスは，リスト1,2それぞれ，3000_H，$B000_H$です。

それでは，使い方を説明しましょう。「MACINTO-C」を起動をすると，最初にエディットするアドレスを指定してきます。そこで，入力したいプログラムの先頭アドレスや，入力の続きのアドレスを4桁の16進数で入力してください。

入力が終わると1ブロック（128バイト）分のダンプリストと縦横チェックサム，CRCチェックサムが表示されます。

そこでの，操作方法は，

T……1ブロックページダウン
G……1ブロックページアップ
E……エディットモードに移り，表示されたブロックをエディットする

となります。なお，"E"でエディットモードに移った場合，BREAKキーで通常のモードに戻ります。

そして，起動直後のアドレス入力場面で"P"と入力するとプリンタ印字モードになります。"P"と入力したあとに，印字開始アドレスと終了アドレスを聞いてきますので，それぞれ4桁の16進数でアドレスを入力してください。入力が終わったら，プリンタの準備ができているか確認し，何かキーを押してください。

また，7月号でMSX用S-OS"SWORD"のモニタコマンドの説明が抜けていましたので，表1に掲載します。MSX用S-OS"SWORD"に関するバグ，拡張コマンドの改造が，今月号の154ページ「ごめんなさいのコーナー」に掲載されています。MSXユーザーの方はそちらも参照してください。

表1 MSX用S-OS "SWORD" モニタコマンド

（[ ] は省略可能であることを示す）

#D [<デバイス名>：]
　<デバイス名>で指定されたデバイスのディレクトリを表示する。省略時はデフォルトのディレクトリ。

#DV<デバイス名>：
　<デバイス名>で指定されたデバイスをデフォルトデバイスに変更する

#J<アドレス>
　<アドレス>から始まるプログラムをコールする。サブルーチン中のRET命令でS-OSのモニタにリターンする。

#K<ファイル名>
　<ファイル名>で与えられたファイルを消去する。

#L<ファイル名>[:<ロードアドレス>]
　<ファイル名>で与えられたファイルを<ロードアドレス>へロードする。<ロードアドレス>が省略されたときには，セーブしたときのアドレスへロードする。

#M
　各機種のマシン語モニタのホットスタートへジャンプする。

#N<ファイル名1>：<ファイル名2>
　<ファイル名1>を<ファイル名2>に変更する。なお，<ファイル名2>のデバイス指定は不要。

#S<ファイル名>：<開始番地>：<終了番地> [：<実行番地>]
　<開始番地>から<終了番地>までを<ファイル名>でセーブする。

#ST<ファイル名>：P　または　：R
　<ファイル名>で指定されたファイルにライトプロテクトをかける。そのあとは同ファイルのセーブ，消去ができなくなる。プロテクトを外すにはRを指定。

#W
　画面の40桁，80桁モードを切り替える。

#!
　呼び出されたシステムに戻る。

リスト1（3000_H版）

```
3000 CD 08 33 11 89 32 CD E4 : 85
3008 32 CD ED 32 1A FE 1B CA : 1B
3010 0E 33 21 0C 00 19 EB 1A : 8C
3018 FE 50 CA 94 30 FE 70 20 : 6A
3020 05 3E 50 CA 94 30 CD FF : ED
3028 32 38 D5 22 7D 32 CD E1 : BE
3030 32 21 00 00 CD 05 33 11 : 69
3038 96 32 CD E4 32 CD E1 32 : 8B
3040 CD E1 32 01 0F 08 CD 69 : 2E
3048 31 CD F3 32 28 B2 CD F0 : BA
3050 32 FE 53 28 AB FE 54 20 : C8
3058 0E 2A 7D 32 11 80 00 B7 : 2F
3060 ED 52 22 7D 32 18 DC FE : 02
3068 47 20 0C 2A 7D 32 11 80 : DD
3070 00 19 22 7D 32 18 CC CD : 9B
3078 0B 33 20 0F 2A 7D 32 5D : A3
-----------------------------------
SUM: 87 B5 62 73 E1 92 CA E3 883F

3080 54 13 01 7F 00 36 00 ED : 0A
3088 B0 18 B8 FE 45 20 05 CD : B5
3090 45 32 18 AF FE 50 20 B1 : 5D
3098 CD 08 33 CD EA 32 11 89 : 8B
30A0 32 CD E4 32 CD ED 32 1A : 1B
30A8 FE 1B CA 00 30 21 0C 00 : 40
30B0 19 EB CD FF 32 38 E4 22 : 40
30B8 7D 32 11 BD 32 CD E4 32 : 92
30C0 CD ED 32 1A FE 1B 28 D3 : 1A
30C8 21 0C 00 19 EB CD FF 32 : 2F
30D0 38 E8 E5 ED 5B 7D 32 B7 : B3
30D8 ED 52 E1 38 BE 22 7F 32 : E9
```

```
30E0 11 CA 32 CD E4 32 CD ED : AA
30E8 32 1A FE 1B 28 AD 21 10 : 6B
30F0 00 19 EB 1A E6 DF FE 59 : 3A
30F8 CC E7 32 CD E1 32 2A 7D : 6C
-----------------------------------
SUM: FE 81 D5 0E 63 62 2A 23 6DB0

3100 32 11 80 00 19 EB 2A 7F : 70
3108 32 23 B7 ED 52 38 39 F5 : B1
3110 01 0F 08 CD 6F 31 CD E1 : 33
3118 32 2A 7D 32 11 80 00 19 : B5
3120 22 7D 32 F1 CA 9B 30 CD : 24
3128 F3 32 CA 9B 30 CD F0 32 : A9
3130 FE 20 20 CA CD F0 32 47 : 3E
3138 B7 28 F9 CD F3 32 CA 9B : 2F
3140 30 78 FE 20 20 B8 18 EC : A2
3148 2A 7F 32 ED 5B 7D 32 B7 : 89
3150 ED 52 23 7D 0E FF 0C D6 : CE
3158 08 28 02 30 F9 C6 08 47 : 70
3160 CD 6F 31 CD E1 32 C3 9B : AB
3168 30 21 00 02 CD 05 33 C5 : 1D
3170 C5 21 81 32 36 00 11 82 : 62
3178 32 01 07 00 ED B0 2A 7D : 7E
-----------------------------------
SUM: A4 87 DF CA F8 3F DB 6E BA4A

3180 32 C1 C5 79 B7 28 08 06 : 1E
3188 08 CD F6 31 0D 20 F8 C1 : E2
3190 CD F6 31 3E 2D 06 21 CD : 53
3198 DB 32 10 FB CD E1 32 11 : 09
31A0 B8 32 CD E4 32 21 81 32 : A1
31A8 06 08 CD DE 32 7E 23 CD : 59
31B0 F6 32 10 F6 CD DE 32 C1 : CC
31B8 79 87 87 87 80 47 2A 7D : 7C
31C0 32 56 5A 23 05 28 27 5E : B7
31C8 23 05 28 22 D5 1E 80 D9 : BE
31D0 E1 D9 7E A3 28 01 37 D9 : 14
31D8 ED 6A 30 08 3E 10 AC 67 : F0
31E0 3E 21 AD 6F D9 CB 0B 30 : 5A
31E8 E9 23 10 E6 D9 EB EB CD : 7E
31F0 F9 32 CD E1 32 C9 3E 08 : 1A
31F8 90 F5 E5 21 81 32 E3 1E : 3F
-----------------------------------
SUM: E2 B2 CC 69 14 FB F4 7C 7DB6

3200 00 CD F9 32 CD DE 32 7E : 53
3208 CD F6 32 7E 83 5F 7E 23 : F6
3210 E3 86 77 23 E3 10 ED E3 : C6
3218 E1 F1 B7 28 0C 3D CD DE : A5
3220 32 CD DE 32 CD DE 32 18 : 04
3228 F1 CD DE 32 3E 3A CD DB : EE
3230 32 CD DE 32 7B CD F6 32 : 7F
3238 C3 E1 32 C5 01 0F 08 CD : 80
3240 69 31 C1 18 02 0E 02 61 : E6
3248 2E 05 CD 05 33 CD ED 32 : 24
3250 CD 02 33 4C 0D 1A FE 1B : 8E
3258 C8 CD FF 32 38 DD 13 06 : F4
3260 08 1A FE 20 20 03 13 18 : 8E
3268 F8 CD FC 32 38 CD 77 23 : 92
3270 10 EF 0C C5 01 0F 08 CD : B5
3278 69 31 C1 18 CA 00 00 00 : 3D
-----------------------------------
SUM: 4E 8E AC 20 63 2F F9 10 9DC9

3280 00 00 00 00 00 00 00 00 : 00
3288 00 53 54 41 52 54 20 41 : EF
3290 44 52 53 3D 24 00 41 44 : CF
3298 52 53 20 2B 30 20 2B 31 : 9C
32A0 20 2B 32 20 2B 33 20 2B : 46
32A8 34 20 2B 35 20 2B 36 20 : 55
32B0 2B 37 20 3A 53 55 4D 00 : B1
32B8 53 55 4D 3A 00 45 4E 44 : 06
32C0 20 20 20 41 44 52 53 3D : C7
32C8 24 00 50 52 49 4E 54 45 : F6
32D0 52 20 4F 4E 20 28 59 2F : DF
32D8 4E 29 00 C3 F4 1F C3 F1 : 01
32E0 1F C3 EE 1F C3 E5 1F C3 : 79
32E8 D9 1F C3 D6 1F C3 1A 33 : C0
32F0 C3 D0 1F C3 CD 1F C3 C1 : E5
32F8 1F C3 BE 1F C3 B5 1F C3 : 19
-----------------------------------
SUM: 26 AD DE ED 57 CF 5B 61 391F

3300 B2 1F C3 18 20 C3 1E 20 : CD
3308 C3 11 33 C3 17 33 C3 21 : F8
3310 33 3E 0C CD F4 1F C9 FE : 24
3318 0C C9 ED 5B 76 1F C3 D3 : 48
3320 1F C9                   : E8
-----------------------------------
SUM: D3 00 EF 03 A1 34 6D 12 9358
```

リスト2(B000$_H$版)

```
B000 CD 08 B3 11 89 B2 CD E4 : 85
B008 B2 CD ED B2 1A FE 1B CA : 1B
B010 0E B3 21 0C 00 19 EB 1A : 0C
B018 FE 50 CA 94 B0 FE 70 20 : EA
B020 05 3E 50 CA 94 B0 CD FF : 6D
B028 B2 38 D5 22 7D B2 CD E1 : BE
B030 B2 21 00 00 CD 05 B3 11 : 69
B038 96 B2 CD E4 B2 CD E1 B2 : 0B
B040 CD E1 B2 01 0F 08 CD 69 : AE
B048 B1 CD F3 B2 28 B2 CD F0 : BA
B050 B2 FE 53 28 AB FE 54 20 : 48
B058 0E 2A 7D B2 11 80 00 B7 : AF
B060 ED 52 22 7D B2 18 DC FE : 82
B068 47 20 0C 2A 7D B2 11 80 : 5D
B070 00 19 22 7D B2 18 CC CD : 1B
B078 0B B3 20 0F 2A 7D B2 5D : A3
-----------------------------------
SUM: 07 35 62 F3 E1 92 CA 63 B4AF

B080 54 13 01 7F 00 36 00 ED : 0A
B088 B0 18 B8 FE 45 20 05 CD : B5
B090 45 B2 18 AF FE 50 20 B1 : DD
B098 CD 08 B3 CD EA B2 11 89 : 8B
B0A0 B2 CD E4 B2 CD ED B2 1A : 9B
B0A8 FE 1B CA 00 B0 21 0C 00 : C0
B0B0 19 EB CD FF B2 38 E4 22 : C0
B0B8 7D B2 11 BD B2 CD E4 B2 : 12
B0C0 CD ED B2 1A FE 1B 28 D3 : 9A
B0C8 21 0C 00 19 EB CD FF B2 : AF
B0D0 38 E8 E5 ED 5B 7D B2 B7 : 33
B0D8 ED 52 E1 38 BE 22 7F B2 : 69
B0E0 11 CA B2 CD E4 B2 CD ED : AA
B0E8 B2 1A FE 1B 28 AD 21 10 : EB
B0F0 00 19 EB 1A E6 DF FE 59 : 3A
B0F8 CC E7 B2 CD E1 B2 2A 7D : 6C
-----------------------------------
SUM: FE 81 D5 8E E3 E2 2A A3 7F4A

B100 B2 11 80 00 19 EB 2A 7F : F0
B108 B2 23 B7 ED 52 38 39 F5 : 31
B110 01 0F 08 CD 6F B1 CD E1 : B3
B118 B2 2A 7D B2 11 80 00 19 : B5
B120 22 7D B2 F1 CA 9B B0 CD : 24
B128 F3 B2 CA 9B B0 CD F0 B2 : 29
B130 FE 20 20 CA CD F0 B2 47 : BE
B138 B7 28 F9 CD F3 B2 CA 9B : AF
B140 B0 78 FE 20 20 B8 18 EC : 22
B148 2A 7F B2 ED 5B 7D B2 B7 : 89
B150 ED 52 23 7D 0E FF 0C D6 : CE
B158 08 28 02 30 F9 C6 08 47 : 70
B160 CD 6F B1 CD E1 B2 C3 9B : AB
B168 B0 21 00 02 CD 05 B3 C5 : 1D
B170 C5 21 81 B2 36 00 11 82 : E2
B178 B2 01 07 00 ED B0 2A 7D : FE
-----------------------------------
SUM: A4 07 5F CA 78 BF DB EE C42D

B180 B2 C1 C5 79 B7 28 08 06 : 9E
B188 08 CD F6 B1 0D 20 F8 C1 : 62
B190 CD F6 B1 3E 2D 06 21 CD : D3
B198 DB B2 10 FB CD E1 B2 11 : 09
B1A0 B8 B2 CD E4 B2 21 81 B2 : 21
B1A8 06 08 CD DE B2 7E 23 CD : D9
B1B0 F6 B2 10 F6 CD DE B2 C1 : CC
B1B8 79 87 87 87 80 47 2A 7D : 7C
B1C0 B2 56 5A 23 05 28 27 5E : 37
B1C8 23 05 28 22 D5 1E 80 D9 : BE
B1D0 E1 D9 7E A3 28 01 37 D9 : 14
B1D8 ED 6A 30 08 3E 10 AC 67 : F0
B1E0 3E 21 AD 6F D9 CB 0B 30 : 5A
B1E8 E9 23 10 E6 D9 EB EB CD : 7E
B1F0 F9 B2 CD E1 B2 C9 3E 08 : 1A
B1F8 90 F5 E5 21 81 B2 E3 1E : BF
-----------------------------------
SUM: E2 B2 4C E9 94 7B F4 FC E2F6

B200 00 CD F9 B2 CD DE B2 7E : 53
B208 CD F6 B2 7E 83 5F 7E 23 : 76
B210 E3 86 77 23 E3 10 ED E3 : C6
B218 E1 F1 B7 28 0C 3D CD DE : A5
B220 B2 CD DE B2 CD DE B2 18 : 84
B228 F1 CD DE B2 3E 3A CD DB : 6E
B230 B2 CD DE B2 7B CD F6 B2 : FF
B238 C3 E1 B2 C5 01 0F 08 CD : 00
B240 69 B1 C1 18 02 0E 02 61 : 66
B248 2E 05 CD 05 B3 CD ED B2 : 24
B250 CD 02 B3 4C 0D 1A FE 1B : 0E
B258 C8 CD FF B2 38 DD 13 06 : 74
B260 08 1A FE 20 20 03 13 18 : 8E
B268 F8 CD FC B2 38 CD 77 23 : 12
B270 10 EF 0C C5 01 0F 08 CD : B5
B278 69 B1 C1 18 CA 00 00 00 : BD
-----------------------------------
SUM: 4E 8E 2C 20 E3 2F F9 10 0269

B280 00 00 00 00 00 00 00 00 : 00
B288 00 53 54 41 52 54 20 41 : EF
B290 44 52 53 3D 24 00 41 44 : CF
B298 52 53 20 2B 30 20 2B 31 : 9C
B2A0 20 2B 32 20 2B 33 20 2B : 46
B2A8 34 20 2B 35 20 2B 36 20 : 55
B2B0 2B 37 20 3A 53 55 4D 00 : B1
B2B8 53 55 4D 3A 00 45 4E 44 : 06
B2C0 20 20 20 41 44 52 53 3D : C7
B2C8 24 00 50 52 49 4E 54 45 : F6
B2D0 52 20 4F 4E 20 28 59 2F : DF
B2D8 4E 29 00 C3 F4 1F C3 F1 : 01
B2E0 1F C3 EE 1F C3 E5 1F C3 : 79
B2E8 D9 1F C3 D6 1F C3 1A B3 : 40
B2F0 C3 D0 1F C3 CD 1F C3 C1 : E5
B2F8 1F C3 BE 1F C3 B5 1F C3 : 19
-----------------------------------
SUM: 26 AD DE ED 57 CF 5B E1 6D0D

B300 B2 1F C3 18 20 C3 1E 20 : CD
B308 C3 11 B3 C3 17 B3 C3 21 : F8
B310 B3 3E 0C CD F4 1F C9 FE : A4
B318 0C C9 ED 5B 76 1F C3 D3 : 48
B320 1F C9                   : E8
-----------------------------------
SUM: 53 00 6F 03 A1 B4 6D 12 B375
```

▶ 全 機 種 共 通 システムインデックス ◀

＊以下のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんのでご注意ください。

1986

■86年1月号

第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ

■86年2月号

第15部　S-OS“SWORD”
第16部　Prolog-85入門(1)

■86年3月号

第17部　magiFORTH発表
連載　Prolog-85入門(2)

■86年4月号

第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)

■86年5月号

第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH

■86年6月号

第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ＆エディタ
第24部　“SWORD”2000 QD
連載　対話で学ぶmagiFORTH
特別付録　PC-8801版S-OS“SWORD”

■86年7月号

第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版S-OS“SWORD”

■86年8月号

第26部　対局五目並べ
第27部　MZ-2500版S-OS“SWORD”

■86年9月号

第28部　FuzzyBASIC発表
連載　明日に向かってmagiFORTH

■86年10月号

第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタDREAM
第31部　FuzzyBASIC料理法<1>

■86年11月号

第32部　パズルゲームHOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC料理法<2>

■86年12月号

第34部　CASL & COMET
連載　FuzzyBASIC料理法<3>

1987

■87年1月号

第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC料理法<4>

■87年2月号

第36部　アドベンチャーゲームMARMALADE
第37部　テキアベ作成ツールCONTEX

■87年3月号

第38部　魔法使いはアニメがお好き
第39部　アニメーションツールMAGE
付録　“SWORD”再掲載とMAGICの標準化

■87年4月号

第40部　INVADER GAME
第41部　TANGERINE

■87年5月号

第42部　S-OS“SWORD”変身セット
第43部　MZ-700用“SWORD”をQD対応に

■87年6月号

インタラプト　コンパイラ物語
第44部　FuzzyBASICコンパイラ
第45部　エディタアセンブラZEDA-3

■87年7月号

第46部　STORY MASTER

■87年8月号

第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージJACKWRITE
特別付録　FM-7/77版S-OS“SWORD”

■87年9月号

第49部　リロケータブル逆アセンブラInside-R
特別付録　PC-8001/8801版S-OS“SWORD”

■87年10月号

第50部　tiny CORE WARS
第51部　FuzzyBASICコンパイラの拡張
第52部　X1turbo版S-OS“SWORD”

■87年11月号

序論　神話のなかのマイクロコンピュータ
付録　S-OSの仲間たち
第53部　もうひとつのFuzzyBASIC入門
第54部　ファイルアロケータ＆ローダ
インタラプト　S-OSこちら集中治療室
第55部　BACK GAMMON

■87年12月号

第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo版“SWORD”アフターケア
ラインプリントルーチン
特別付録　PASOPIA7版S-OS“SWORD”

1988

■88年1月号

第58部　FuzzyBASICコンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正

■88年2月号

第59部　シューティングゲームELFES

■88年3月号

第60部　構造型コンパイラ言語SLANG

■88年4月号

第61部　デバッギングツールTRADE
第62部　シミュレーションウォーゲームWALRUS

■88年5月号

第63部　シューティングゲームELFESⅡ
第64部　地底最大の作戦

■88年6月号

第65部　構造化言語SLANG入門(1)
第66部　Lisp-85用NAMPAシミュレーション

■88年7月号

第67部　マルチウィンドウドライバMW-1
連載　構造化言語SLANG入門(2)

■88年8月号

第68部　マルチウィンドウエディタWINER

■88年9月号

第69部　超小型エディタTED-750
第70部　アフターケアWINERの拡張

■88年10月号

第71部　SLANG用ファイル入出力ライブラリ
第72部　シューティングゲームMANKAI

■88年11月号

第73部　シューティングゲームELFESⅣ

■88年12月号

第74部　ソースジェネレータSOURCERY

1989

■89年1月号

第75部　パズルゲームLAST ONE
第76部　ブロックゲームFLICK

■89年2月号

第77部　高速エディタアセンブラREDA
特別付録　X1版S-OS“SWORD”<再掲載>

■89年3月号

第78部　Z80用浮動小数点演算パッケージSOR
OBAN

■89年4月号

第79部　SLANG用実数演算ライブラリ

■89年5月号

第80部　ソースジェネレータRING

■89年6月号

第81部　超小型コンパイラTTC

■89年7月号

第82部　TTC用パズルゲームTICBAN

■89年8月号

第83部　CP/M用ファイルコンバータ

■89年9月号

第84部　生物進化シミュレーションBUGS

■89年10月号

第85部　小型インタプリタ言語TTI

■89年11月号

第86部　TTI用パズルゲームPUSH BON!

■89年12月号

第87部　SLANG用リダイレクションライブラリDIO.LIB

1990

■90年1月号

第88部　SLANG用ゲームWORM KUN
特別付録　再掲載SLANGコンパイラ

■90年2月号

第89部　超小型コンパイラTTC++

■90年3月号

第90部　超多機能アセンブラOHM-Z80

■90年4月号

第91部　ファジィコンピュータシミュレーションI-MY

■90年5月号

第92部　インタプリタ言語STACK

■90年6月号

第93部　リロケータブルフォーマットの取り決め
第94部　STACK用ゲームSQUASH!
第95部　X68000対応S-OS“SWORD”
特別付録　PC-286対応S-OS“SWORD”

■90年7月号

第96部　リロケータブルアセンブラWZD

■90年8月号

第97部　リンカWLK

■90年9月号

第98部　BILLIARDS

■90年10月号

第99部　ライブラリアンWLB

■90年11月号

第100部　タブコード対応エディタEDC-T

■90年12月号

第101部　STACKコンパイラ

1991

■91年1月号

第102部　ブロックアクションゲームCOLUMNS

■91年2月号

第103部　ダイスゲームKISMET

■91年3月号

第104部　アクションゲームMUD BALLIN'

■91年4月号

第105部　SLANG用カードゲームDOBON

■91年5月号

第106部　実数型コンパイラ言語REAL

■91年6月号

第107部　Small-C処理系の移植

■91年7月号

第108部　REALソースリスト編

■91年8月号

第109部　Small-Cライブラリの移植

■91年9月号

第110部　SLANG用NEWファイル出力ライブラリ

■91年10月号

第111部　Small-C活用講座（初級編）

■91年11月号

第112部　Small-C活用講座（応用編）
第113部　MORTAL

■91年12月号

第114部　Small-C SLANGコンパチ関数

1992

■92年1月号

第115部　LINER

■92年2月号

第116部　シミュレーションゲームPOLANYI

■92年3月号

第117部　カードゲームKLONDIKE

■92年4月号

第118部　オプティマイザO80実践Small-C講座(1)

■92年5月号

第119部　COMMAND.OBJ実践Small-C講座(2)

■92年6月号

第120部　COMMAND.OBJ2実践Small-C講座(3)

■92年7月号

第121部　関数リファレンス実践Small-C講座(4)

■92年8月号

第122部　ワイルドカード実践Small-C講座(5)
第123部　グラフィックライブラリ GRAPH.LIB

■92年9月号

第124部　O-EDIT&MODCNV

■92年10月号

第125部　SLENDER HUL実践Small-C講座(6)

■92年11月号

第126部　EDIT実践Small-C講座(7)

■92年12月号

第127部　MAKE実践Small-C講座(8)

Oh!X Books改訂版９月上旬出来予定

Z-MUSICシステムver.2.0

ながらくお待たせしました。ついにZ-MUSICシステムの正式バージョンアップ版が登場します。X68000用の音源ドライバとしてさらに使いやすく高機能なものになりました。

●ver.1.0/1.1からのバージョンアップ内容
PCM8対応AD PCM同時発音８声音量可変
モジュレーション用波形メモリ搭載
PCMバンクに対応
ステップエディット系コマンド追加
X68030完全対応ユニバーサルバージョン
RS-232C対応版収録
POLYPHON対応版収録
再生専用機能縮小版収録
Cコンパイラ用ライブラリ完成
AD PCM加工機能強化
さらにクオリティを高めたAD PCMデータ
リファレンスマニュアル＆導入マニュアル
その他，全面的改良＆機能整備

もちろん，全ソースプログラム付属＆ライセンスフリー

5″2HD 6枚組　　予価5,000円（税込）

以下続刊（するといいな）
・Z-MUSICサポートツール集
・Oh!X LIVEデータ集

Oh!X

LIVE in '93

X68000・Z-MUSIC +PCM8用 ©SEGA OutRunより

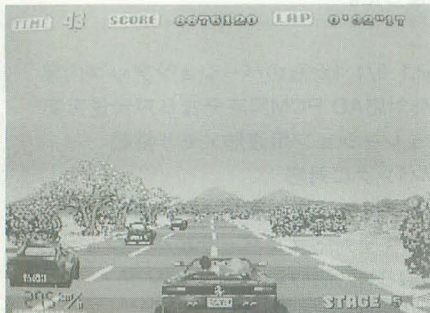
SPLASH WAVE

Sindo Noriyuki 進藤 慶到

暑い季節がやってきました。やはりここは，お気に入りのゲームに燃えて身もココロもアツくならなくちゃ!? ということで，1993年5月号に掲載したOutRun，リクエストにお応えして2曲目の掲載です。ゲームだけじゃなく入力もしてみてね。

やっぱりアウトランだぜ

内蔵音源派のみなさまこんにちは。5月号の「MAGICAL SOUND SHOWER」はいかがでしたか。あまり似ていなくてガッカリされた方，ごめんなさい。ご意見やリクエストをお寄せくださったみなさん，ありがとうございます。残りを作ってくれという声が予想どおり多いです。「OutRun」ってやっぱり偉大だったのね……。

そういえば，みなさんはもうセガの新作ゲーム「OutRunners」をプレイしましたか？ 私は，音楽の評判がいいので期待していたのですが，結局「OutRun」の持つ音楽性の高さを再認識させられたような気がしました。いいものはいつまでも色褪せることがないのだな，うむ。

OutRun

で，今回はCDの順番に従い「SPLASH WAVE」を作ってみました。この曲も人気が非常に高いようですね。まあ，1回聴けば，それにも十分うなずけるというもの。特に後半のメロディがオクターブアップして盛り上がってくるところなんか最高です。この曲は私が「OutRun」をプレイするときにいつも鳴らしていた思い出深い曲なので，よりいっそう愛を込めて作りました。ぜひ聴いてね。

演奏にはいつものとおりZ-MUSICシステムとPCM8.Xが必要です。それから，5月号のリストを入力してくれた人ならわかることと思いますが，恒例となった(?)「スーパーハングオン」のドラムデータを分解するBASICリストも再掲載しますので，お持ちの方は用意してください。臨場感の高いサウンドが再現できるでしょう（逆に標準AD PCMデータではかなり情けない）。

それにしても，私はいったいいくつこの曲を作ってきたのでしょうか……。使用機種やドライバが変わるたびにネタにしてきましたからね。MZ-2000のOPNボードで演奏したときなんか，ドラムマシンを使用したんですが，互いにテンポを合わせる術がなく，ツマミを必死に回しながら自らテンポを合わせたという苦い思い出があります。MIDI楽器やPCM8.Xが自由に使える現在では考えられないようなことですね。

さて，ZPDデータ作成の注意点をもう一度。「スーパーハングオン」のゲームディスクAの「¥BGM¥SPHSND.MOP」というファイルとBASICリストを同じディレクトリにコピーして，BASICリストを実行してください。いくつかのPCMファイルに分割されるので，これを使ってZPDを作るわけです。「スーパーハングオン」がない場合はいつものようにZPDを作ります。

こうしてできたZPDデータは，5月号の「MAGICAL SOUND SHOWER」と共有できます。ファイル名かZPD登録行を書き換えて，うまく対応してください。たとえばファイル名を「OUTRUN.ZPD」として，曲中では.ADPCM_BLOCK_DATA=OUTRUNとすればいいわけです。

さて曲のデキはどうでしょうか。ドラムパートにはかなり神経を使いました。音色は「MAGICAL SOUND SHOWER」よりも楽だったのですが，全体を通してみるとやっぱり疲れましたね。ご意見待ってます。

それでは，またお会いしましょう。「LAST WAVE」もできているので，リクエストもお待ちしております。（進藤慶到）

リスト1 SPLASH WAVE

```
 1: .comment -OUT RUN- SPLASH WAVE (C)SEGA  by ENG (+PCM8)
 2:
 3: / for ZMUSIC.X + PCM8.X
 4:
 5: /-----------------------------------------------------------
 6: / TRACK SETUP
 7:
 8: (i)
 9:
10: / OPM & ADPCM
11:
12: (m1,3000)(aFm1,1)
13: (m2,3000)(aFm2,2)
14: (m3,3000)(aFm3,3)
15: (m4,3000)(aFm4,4)
16: (m5,3000)(aFm5,5)
17: (m6,3000)(aFm6,6)
18: (m7,3000)(aFm7,7)
19: (m8,3000)(aFm8,8)
20:
21: (m09,2000)(aAdpcm,09)
22: (m10,2000)(aAdpcm,10)
23: (m11,2000)(aAdpcm,11)
24: (m12,2000)(aAdpcm,12)
25: (m13,2000)(aAdpcm,13)
26: (m14,2000)(aAdpcm,14)
27:
28: /-----------------------------------------------------------
29: / ADPCM DATA SET
30:
31: .adpcm_block_data = SP_WAVE
32:
33: / 今回作った .ZPD データは、1993年5月号のマジカルサウンド
34: / シャワーで使うこともできます（逆は不可）。
35:
36: / ファイルネームを変えて、ZPDを共有するのがいいと思います。
37:
38: /-----------------------------------------------------------
39: / OPM DATA SET
40:
41: /       AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME  BASS
42: (@1,    29,  9,  7,  9,  2, 24,  1,  0,  3,  0,  0
43:         29, 11, 10,  9,  1,  6,  1,  0,  0,  0,  0
44:         29,  7,  6,  9,  1,  3,  1,  1,  0,  0,  0
45:         29,  7,  6,  9,  1,  1,  1,  2,  0,  0,  0
46: /       AL  FB  OM PAN
47:          5,  6)
48:
49: /       AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME  MAIN L
50: (@2,    18,  8,  7,  8,  3, 22,  0,  1,  1,  0,  0
51:         18,  8,  6, 10,  7,  3,  0,  2,  2,  0,  0
52:         18,  8,  6, 10,  7,  3,  0,  2,  2,  0,  0
```

```
 53:          18,  8,  6, 10,  7, 10,  0,  4,  5,  0,  0
 54: /         AL  FB  OM PAN
 55:            5,  6)
 56:
 57: /         AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME  SUB
 58: (@3,      30, 22,  9,  9,  1,  6,  0,  1,  7,  0,  0
 59:           31,  0,  9,  7,  0,  3,  0,  1,  1,  0,  0
 60:           31, 15,  9,  7,  1,  1,  0,  1,  3,  0,  0
 61:           31, 13,  0,  7,  2, 18,  0,  8,  5,  2,  0
 62: /         AL  FB  OM PAN
 63:            6,  0)
 64:
 65: /         AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME  CHORD
 66: (@4,      31,  7,  3,  9,  3, 32,  0,  2,  1,  0,  0
 67:           28,  3,  3, 10,  1, 32,  0,  1,  2,  0,  0
 68:           28,  0,  0, 10,  0, 16,  0,  1,  0,  0,  0
 69:           25,  0,  0, 15,  0,  3,  0,  1,  1,  0,  0
 70: /         AL  FB  OM PAN
 71:            0,  5)
 72:
 73: /         AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2 AME  MAIN H
 74: (@5,      18,  8,  6,  8,  3, 22,  0,  1,  1,  0,  0
 75:           18,  8,  6, 10,  6,  2,  0,  2,  0,  0,  0
 76:           18,  8,  6, 10,  6,  3,  0,  2,  0,  0,  0
 77:           17,  7,  6, 10,  6, 11,  1,  1,  2,  0,  0
 78: /         AL  FB  OM PAN
 79:            5,  6)
 80:
 81: /-----------------------------------------------------------
 82: / MML DATA SET
 83:
 84: (o79)
 85:
 86: /-------------------------------
 87: / BASS
 88:
 89: (t1) @v0r*4L16@1o2@v113@k01q8
 90: (t1) aa<a>ara<a>a-a-a-<a->a-<a->a-<a->
 91: (t1) g8g<g>g<g-ga>d8d<d&>d<d+&>d<e&>d
 92: (t1) |:aa<a>a<<c>a>a<a-8>a-<a->a-<ba->a-g&
 93: (t1) g<g>gg<bagd8>d<d>d<e>d<g->d:|
 94: (t1) ra<a>a<g>a<a>a-8a-<a->a-<g>a-<a->g&gg<g>g<g-ga>d<
 95: (t1) _2{da<d>d+a+<d+>eb<e>f<cf>}2
 96: (t1) {f+<c+f+>g<dg>g+<d+g+>a<ea>}2>~
 97: (t1) |:|:
 98: (t1) |:8q8g<@q6g>:||:8q8d<@q6d>:|
 99: (t1) |:8q8a<@q6a>:||:6q8e<@q6e>:|q8eeef&
100: (t1) |:8q8f<@q6f>:||:7q8c<@q6c>:|q8d<@q6d>
101: (t1) |:8q8e-<@q6e->:||:3q8d<@q6d>:|q8dd&
102: (t1) q8d<@q6d>q8e-<@q6e->q8e<@q6e>q8g-<@q6g->:|q8<
103: (t1) rfcdfrcdffcdfcdfre>b<der>b<de>b<dere>b<d
104: (t1) rfcdfrcdrfcdfcdfre>b<de>b<dee<e>ee<d>eb<c>
105: (t1) |:rfcdfcdfrfcdfcdfre>b<de>b<dere|e-egeae:|
106: (t1) >b<d{ee-d&}4
107: (t1) d8<d8>dd<d>g8g<g>g<g>gg
108: (t1) c8c<c>cgag>ara<d-e>a<a>a<
109: (t1) ed<d>d<d8d>dg8g<g>g<g->g<e>
110: (t1) c8c<c>cc>ga-a8<a>a<a>a<a>a<d&
111: (t1) dd<d>d8<dd>g8g<g>g<e>g<d>e&
112: (t1) ee<e>e<e>e>ga8a<d->a<e>a<gd&
113: (t1) dd<d>d<d8>dg8g<g>g<g>dd-c&
114: (t1) cc<c>crc<c>cr|@q6d*17d.d.r*31
115: (t1) :|q8_1ddddddd~
116: (t1) r*768o2
117: (t1) aa<a>ara<a>a-a-a-<a->a-<a->a-<a->
118: (t1) g8g<g>g<g-ga>d8d<d&>d<d+&>d<e&>d
119: (t1) |:aa<a>a<<c>a>a<a-8>a-<a->a-<ba->a-g&
120: (t1) g<g>gg<bagd8>d<d>d<e>d<g->d:|
121: (t1) ra<a>a<g>a<a>a-8a-<a->a-<g>a-<a->g&gg<g>g<g-ga>d<
122: (t1) _2{da<d>d+a+<d+>eb<e>f<cf>}2
123: (t1) {f+<c+f+>g<dg>g+<d+g+}*72r8>~
124: (t1) d8<d8>dd<d>g8g<g>g<g>gg
125: (t1) c8c<c>cgag>ara<d-e>a<a>a<
126: (t1) ed<d>d<d8d>dg8g<g>g<g->g<e>
127: (t1) c8c<c>cc>ga-a8<a>a<a>a<a>a<d&|:
128: (t1) dd<d8>dd<d>g8g<g>g<g>gg
129: (t1) c8c<c>cgag>ara<d-e>a<a>a<
130: (t1) ed<d>d<d8d>dg8g<g>g<g->g<e>
131: (t1) c8c<c>cc>ga-a8<a>a<a>a<a>a<d&:|
132: (t1) dd<d8>dd<d>g8g<g>g<e>g<d>e&
133: (t1) ee<e>e<e>e>ga8a<d->a<e>a<gd&
134: (t1) dd<d>d<d8>dg8g<g>g<g>dd-c&
135: (t1) cc<c>crc<c>cr2>
136: (t1) |:aa<a>ara<a>a-a-a-<a->a-<a->a-<a->
137: (t1) g8g<g>g<g-ga>d8d<d&>d<d+&>d<e&>d
138: (t1) aa<a>a<<c>a>a<a-8>a-<a->a-<ba->a-g&
139: (t1) g<g>gg<bagd8>d<d>d<e>d<g->d
140: (t1) :|aar*1350
141:
142: /-------------------------------
143: / CHORD
144:
145: (t2) @v0r*4L16@4o3@v112@k-1q8
146: (t2) |:2<cc_20c_cr8~~cc_c_cr4~~cc_c_cr4~~c>b^4.b8:|
147: (t2) |:<cc_c_cr8~~cc_c_cr4~~cc_c_cr4~~c>b
148: (t2) |_b~b_b~b_b~bb_b~:|_b_b@v0r2..
149: (t2) |:|:@v112
150: (t2) |:bb_20b_br8~~bb_b_br8~~bb_b_b~~:|
151: (t2) |:<cc_c_cr8~~cc_c_cr8~~cc_c_c>~~
152: (t2) bb_b_br8~~bb_b_br8~~bb_b_b~~:|
153: (t2) g-g-_g-_g-r8~~g-g-_g-_g-r8~~g-g-_g-_g-<~~
154: (t2) dd_d_dr8~~dd4..&_d_d
155: (t2) :|r@q1@v112|:4
156: (t2) e_e_e~~e_e_er~~e_e_e~~e*13_e*11~e8. _e~d_d_d~~d_d_d~~
157: (t2) |1rd_d_d~~d4&_d_d~~:|
158: (t2) |2rd_d_dr8~~d8.&_d~:|
159: (t2) |3rd_d_d~~d4&_d_d~~:|
160: (t2) |4d_d~d_d_dq8~~{d~1ec&}4:|
161: (t2) c*84>b2b2a2<c2>b2b^2& _25b~a_b~<d-_>a~<e_d-~
162: (t2) c2>b2<d2d-2c2>b2b^2&_20b<|~<d_d*5~d_d32~d_d_dr*13>>
163: (t2) :|
164: (t2) q8<<@v111dddddd8
165: (t2) _20d_d@v0r*744@v112>>
166: (t2) |:2<cc_c_cr8~~cc_c_cr4~~cc_c_cr4~~c>b^4.b8:|
167: (t2) |:<cc_c_cr8~~cc_c_cr4~~cc_c_cr4~~c>b
168: (t2) |_b~b_b~b_b~bb_b~:|_b_br*156~41<c&
169: (t2) c*84>b2b2a2<c2>b2b^2& _25b~a_b~<d-_>a~<e_d-~26<
170: (t2) c2>b2<d2d-4e_20e~e8c4..c>b4.b_b~b^2&
171: (t2) _b~<d-d-d-d-d-d-
172: (t2) |:c2>b2<d2d-4..d-c2>b2&|`b2<d-
173: (t2) _25d-~d-_d-~d-_d-~d-_d-~:|>_20b~b2 _30b_br4.>@v113
174: (t2) |:4<cc_20c_cr8~~cc_c_cr4~~cc_c_cr4~~c>b^4.b8:|
175: (t2) ~1<<cc_23c_c
176:
177: (t3) @v0r*4L16@4o3@v111p1@k-1q8
178: (t3) |:2bb_20b_br8~~bb_b_br4~~bb_b_br4~~ba^4.a8:|
179: (t3) |:bb_b_br8~~bb_b_br4~~bb_b_br4~~ba
180: (t3) |_a~a_a~a_a~ag_g~:|_a_a@v0r2..
181: (t3) |:|:@v111
182: (t3) |:aa_a_ar8~~aa_a_ar8~~aa_a_a~~:|
183: (t3) bb_b_br8~~bb_b_br8~~bb_b_b~~
184: (t3) gg_g_gr8~~gg_g_gr8~~gg_g_g~~
185: (t3) aa_a_ar8~~aa_a_ar8~~aa_a_a~~
186: (t3) gg_g_gr8~~gg_g_gr8~~gg_g_g~~
187: (t3) b-b-_b-_b-r8~~b-b-_b-_b-r8~~b-b-_b-_b-~~
188: (t3) aa_a_ar8~~aa4..&_a_a
189: (t3) :|r<@q1@v111|:4
190: (t3) c_c_c~~c_c_cr~~c_c_c~~c*13_c*11~c8.&_c~
191: (t3) |1>b_b_b~~b_b_br~~b_b_b~~b4&_b_b~~<:|
192: (t3) |2>b_b_b~~b_b_br~~b_b_br8~~b8.&_b~<:|
193: (t3) |3 c_c_c~~c_c_cr~~c_c_c~~c4&_c_c~~:|
194: (t3) |4>b_b_b~~b_b_b~~b_b~b_b_b~~q8{b~1<d->a&}4:|
195: (t3) a*84g2g2g4..ga2g2g^2& _25g~g_g~a_g~<c+>_a~
196: (t3) a2g2b2a2a2g2g^2&_20g<|~b_b*5~b_b32~b_b_br*13>
197: (t3) :|
198: (t3) q8@v111bbb~1bbb8
199: (t3) _20b_b@v0r*744@v111
200: (t3) |:2bb_b_br8~~bb_b_br4~~bb_b_br4~~ba^4.a8:|
201: (t3) |:bb_b_br8~~bb_b_br4~~bb_b_br4~~ba
202: (t3) |_a~a_a~a_a~aa_a~:|_a_ar*156~41a&
203: (t3) a*84g2g2g4..ga2g2g^2& _25g~g_g~a_g~<c+>_a~26<
204: (t3) a2g2b2a4a_20a~a8a4..ag4.g_g~g^2&_g~aaaaaa
205: (t3) |:a2g2b2a4..aa2g2&|`g2a
206: (t3) _25a~a_a~a_a~a_a~:|_20g~g2& _30g_gr4.>@v112
207: (t3) |:4bb_20b_br8~~bb_b_br4~~bb_b_br4~~ba^4.a8:|
208: (t3) ~1<ba_23a_a
209:
210: (t4) @v0r*4L16@4o3@v111p2@k-1q8
211: (t4) |:2aa_20a_ar8~~aa-_a-_a-r4~~a-g_g_gr4~~gg-^4.g8:|
212: (t4) |:aa_a_ar8~~aa-_a-_a-r4~~a-g_g_gr4~~gg-
213: (t4) |_g-~g-_g-~g-_g-~g-g_g~:|_g-_g-@v0r2..
214: (t4) |:|:@v111
215: (t4) gg_g_gr8~~gg_g_gr8~~gg_g_g~~
216: (t4) ff_f_fr8~~ff_f_fr8~~ff_f_f~~
217: (t4) aa_a_ar8~~aa_a_ar8~~aa_a_a~~
218: (t4) g-g-_g-_g-r8~~g-g-_g-_g-r8~~g-g-_g-_g-~~
219: (t4) gg_g_gr8~~gg_g_gr8~~gg_g_g~~
220: (t4) ee_e_er8~~ee_e_er8~~ee_e_e~~
221: (t4) b-b-_b-_b-r8~~b-b-_b-_b-r8~~b-b-_b-_b-
222: (t4) ~~gg_g_gr8~~gg-4..&_g-_g-
223: (t4) :|r@q1@v111|:4
224: (t4) a_a_a~~a_a_ar~~a_a_a~~a*13_a*11~a8.& _a~g_g_g~~g_g_g~~
225: (t4) |1rg_g_g~~g4&_g_g~~:|
226: (t4) |2rg_g_gr8~~g8.&_g~:|
227: (t4) |3rg_g_g~~g4&_g_g~~:|
228: (t4) |4g_g~g_g_gq8~~{g~1b-f&}4:|
229: (t4) f*84f2e2e2f2f2e^2& _25e~e_e~g_e~a_g~
230: (t4) f2f2g2g4..gf2f2e^2&_20e|~<g-_g-*5~g-_g-32~g-_g-_g-r*13
231: (t4) >:|
232: (t4) q8@v111g-g-g-g-~1g-g-8
233: (t4) _20g-_g-@v0r*744@v111
234: (t4) |:2aa_20a_ar8~~aa-_a-_a-r4~~a-g_g_gr4~~gg-^4.g8:|
235: (t4) |:aa_a_ar8~~aa-_a-_a-r4~~a-g_g_gr4~~gg-
236: (t4) |_g-~g-_g-~g-_g-~g-g_g~:|_g-_g-r*156~41f&
237: (t4) f*84f2e2e2f2f2e^2& _25e~e_e~g_e~<a_>g~26<
238: (t4) f2f2g2g4g_20g~g8f4..ff4.f_f~e^2&_e~gggggg
239: (t4) |:f2f2g2g4..gf2f2&|`e2g
240: (t4) _25g~g_g~g_g~g_g~:|_20f~e2& _30e_er4.>@v112
241: (t4) |:4aa_20a_ar~~raa-_a-_a-r8.~~ra-g_g_gr8.~~rgg-^4.g8:|
242: (t4) ~1<aa_23a_a
243:
244: (t5) @v0r*4L16@4o3@v112@k-1q8
245: (t5) |:2ee_20e_er8~~ee_e_er4~~ee_e_er4~~ed^4.e8:|
246: (t5) |:ee_e_er8~~ee_e_er4~~ee_e_er4~~ed
247: (t5) |_d~d_d~d_d~dd_d~:|_d_d@v0r2..
248: (t5) |:|:@v112
249: (t5) |:dd_d_dr8~~dd_d_dr8~~dd_d_d~~:|
250: (t5) |:ee_e_er8~~ee_e_er8~~ee_e_e~~:|
251: (t5) ff_f_fr8~~ff_f_fr8~~ff_f_f~~
252: (t5) |:cc_c_cr8~~cc_c_cr8~~cc_c_c~~:|
253: (t5) dd_d_dr8~~dd4..&_d_d
254: (t5) :|r@q1@v112|:4
255: (t5) f_f_f~~f_f_fr~~f_f_f~~f*13_f*11~f8.& _f~e_e_e~~e_e_e~~
256: (t5) |1re_e_e~~e4&_e_e~~:|
257: (t5) |2re_e_er8~~e8.&_e~:|
258: (t5) |3re_e_e~~e4&_e_e~~:|
259: (t5) |4e_e~e_e_e~~q8{e~1g-d&}4:|
260: (t5) d*84d2c2d-2d2d2c^2& _25c~d-_c~e_d-~g_e~
261: (t5) d2d2e2e2d2d2c^2&_20c|~<d_d*5~d_d32~d_d_dr*13>
262: (t5) :|
```

```
263: (t5) q8@v112ddddd¯1d8
264: (t5) _20d_d@v0r*744@v112
265: (t5) |:2ee_20e_er8¯¯ee_e_er4¯¯ee_e_er4¯¯ed^4.e8:|
266: (t5) |:ee_e_er8¯¯ee_e_er4¯¯ee_e_er4¯¯ed
267: (t5) |_d¯d_d¯d_d¯dd_d¯:|_d_dr*156¯41d&
268: (t5) d*84d2c2d-2d2d2c^2& _25c¯d-_c¯<e_>d-¯<g_e¯26
269: (t5) d2d2e2e4e_20e¯e8d4..dd4.d_d¯c^2&_c¯eeeeee
270: (t5) |:d2d2e2e4..ed2d2&|`c2e
271: (t5) _25e¯e_e¯e_e¯e_e¯:|_20d¯c2& _30c_cr4.>@v113
272: (t5) |:4ee_20e_er¯¯ree_e_er8.¯¯ree_e_er8.¯¯red^4.e8:|
273: (t5) ¯1<ee_23e_e
274:
275: /--------------------------------
276: / MAIN
277:
278: (t6) @v0r*4L16@2o4@v108@k02q8
279: (t6) |:3
280: (t6) p2eb<p3_1e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
281: (t6) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da|<p3_cd>¯:|
282: (t6) p3_ga¯
283: (t6) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
284: (t6) p2da<p3_d>d¯p1a<d>p3_d<d@v0r1@k06
285: (t6) |:@v112o3|:
286: (t6) d*69r*1g-*3g*11<c8.>b8a*10b-*3b*11rf*94<e*3f*35e8der>
287: (t6) b*151<c*3c+*3d*23c8>b*67f*3f+*3g*11e4..rf&
288: (t6) f*115g*3g+*3a*23<c>bra8ge*91f*3f+*3g*23barb-&
289: (t6) b-*151<c+*3d*3d+*23>|<d4..c*10c+*3d*83r8>:|
290: (t6) <d*79f*3f+*3g*11g-*85r*23
291: (t6) e*120d8agre*156e>b<de*156agre*185r*14c*3c+*3
292: (t6) d*35cr4r*19c*3c+*3d*11c*10d-*3d*11e8.
293: (t6) d8.cr4r*19c*3c+*3d*11c*10d-*3d*11e*31c*3c+*3
294: (t6) d*35cr*43c*3c+*3d*23cdre8{d>bgg}*144ee12f12g*14
295: (t6) |:5g+*3a*23g*10:||:3a+*3b*23|a*10:|a<cc>b8a8<c+d*22
296: (t6) |:4d+*3e*11|d*22:|d8eg*10g+*3a*11g2rc+rdrerf*7f*3
297: (t6) |:5f+*3g*23|f*10:|f*7g*3
298: (t6) |:3g+*3a*23|g*10:|g<c+>a+a<e8dc+d*22
299: (t6) |:4d+*3e*11|d*22:|d8fe|c*108r2:|c2<c4..r8
300: (t6) o4@v108@k02
301: (t6) |:
302: (t6) p2eb<p3_e>e¯p1b<e>_p3e¯b<p2e>ep3_b<e¯>p1eb<p3_e>b¯
303: (t6) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da<p3_d>a¯:|
304: (t6) |:3
305: (t6) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
306: (t6) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da|<p3_cd>¯:|
307: (t6) p3_ga¯
308: (t6) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
309: (t6) p2da<p3_d>d¯p1a<d>p3_d<dr2@v0
310: (t6) r2@v112@k06o3
311: (t6) a8g*10|:4g+*3a*23g*10:|
312: (t6) |:3a+*3b*23|a*10:|a<cc>b8a8<c+d*22
313: (t6) |:4d+*3e*11|d*22:|d8eg*10g+*3a*11g2rc+rdrerf*7f*3
314: (t6) |:5f+*3g*23|f*10:|f*7g*3
315: (t6) |:3g+*3a*23|g*10:|g<c+>a+a<e8dc+d*22
316: (t6) |:4d+*3e*11|d*22:|d8ebg*103 d*3e-*3e*11d8d-8deg
317: (t6) |:5f8e:||:f*10g-*3g*23:|fg8fb-8a8gfe
318: (t6) |:3d*22e-*3e*11:|dd*10e-*3e*11d8cc*108r>b-r<c+rar
319: (t6) g8|:f*10g-*3|g*23:|g*11gf*22g-*3g*9g-*3g*11fg8
320: (t6) f*7g*3|:3a-*3a*23|g*10:|gr<d->b-agfe
321: (t6) |:5d*10e-*3e*23:|ferc4..r2
322: (t6) o4@v109@k02
323: (t6) |:5
324: (t6) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
325: (t6) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da<p3_cd>¯:|
326: (t6) ¥21
327: (t6) |:5
328: (t6) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
329: (t6) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da<p3_cd>¯:|
330:
331: (t7) @v0r*4L16@3o4@v110@k-3q8
332: (t7) |:3
333: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
334: (t7) p3da<p2d>dp3a<d>p1da<p3d>dp2a<d>p3da|<p1cd>:|p1ga
335: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
336: (t7) p3da<p2d>dp3a<d>p1d<dp3
337: (t7) @4@k-3o3{da<d>d+a+<d+>eb<e>f<cf>}2
338: (t7) {f+<c+f+>g<dg>g+<d+g+>a<ea>}2@2@k00
339: (t7) |:@v111o3|:
340: (t7) d*69r*1g-*3g*11<c8.>b8a*10b-*3b*11rf*94<e*3f*35e8der>
341: (t7) b*151<c*3c+*3d*23c8>b*67f*3f+*3g*11e4..rf&
342: (t7) f*115g*3g+*3a*23<c>bra8ge*91f*3f+*3g*23barb-&
343: (t7) b-*151<c+*3d*3d+*23>|<d4..c*10c+*3d*83r8>:|
344: (t7) <d*79¯1f*3f+*3g*11g-*85r*23_
345: (t7) e*120d8agre*156e>b<de*156agre*185r*14c*3c+*3
346: (t7) d*35cr4r*19c*3c+*3d*11c*10d-*3d*11e8.
347: (t7) d8.cr4r*19c*3c+*3d*11c*10d-*3d*11e*31c*3c+*3
348: (t7) d*35cr*43c*3c+*3d*23cdre8{d>bgg}*144ee12f12g*14
349: (t7) |:5g+*3a*23g*10:||:3a+*3b*23|a*10:|a<cc>b8a8<¯c+d*22
350: (t7) |:4d+*3e*11|d*22:|d8eg*10g+*3a*11g2rc+rdrerf*7f*3
351: (t7) |:5f+*3g*23|f*10:|f*7g*3
352: (t7) |:3g+*3a*23|g*10:|g<c+>a+a<e8dc+d*22
353: (t7) |:4d+*3e*11|d*22:|d8fe|c*108r2:|c2<¯c4..r8
354: (t7) @3o4@v110@k-3
355: (t7) |:
356: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
357: (t7) p3da<p2d>dp3a<d>p1da<p3d>dp2a<d>p3da<p1d>a:|
358: (t7) |:3
359: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
360: (t7) p3da<p2d>dp3a<d>p1da<p3d>dp2a<d>p3da|<p1cd>:|p1ga
361: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
362: (t7) p3da<p2d>dp3a<d>p1d<d
363: (t7) @4@k-3o3{da<d>d+a+<d+>eb<e>f<cf>}2
364: (t7) {f+<c+f+>g<dg>g+<d+g+}*72_14g+_g+@2@v111@k00o3
365: (t7) a8g*10|:4g+*3a*23g*10:|
366: (t7) |:3a+*3b*23|a*10:|a<cc>b8a8<c+d*22
367: (t7) |:4d+*3e*11|d*22:|d8eg*10g+*3a*11g2rc+rdrerf*7f*3
368: (t7) |:5f+*3g*23|f*10:|f*7g*3
369: (t7) |:3g+*3a*23|g*10:|g<c+>a+a<e8dc+d*22
370: (t7) |:4d+*3e*11|d*22:|d8e¯1bg*103 _d*3e-*3e*11d8d-8deg
371: (t7) |:5f8e:|¯|:f*10g-*3g*23:|fg8fb-8a8gf_e
372: (t7) |:3d*22e-*3e*11:|dd*10e-*3e*11d8cc*108r>b-r<c+r¯ar
373: (t7) g8|:f*10g-*3|g*23:|g*11gf*22g-*3g*9g-*3g*11fg8
374: (t7) f*7g*3|:3a-*3a*23|g*10:|gr<d->b-agfe
375: (t7) _|:5d*10e-*3e*23:|ferc4..r2
376: (t7) @3o4@v111@k-3
377: (t7) |:5
378: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
379: (t7) p3da<p2d>dp3a<d>p1da<p3d>dp2a<d>p3da<p1cd>:|
380: (t7) ¥21
381: (t7) |:5
382: (t7) p3eb<p2e>ep3b<e>p1eb<p3e>ep2b<e>p3eb<p1e>b
383: (t7) p3da<p2d>dp3a<d>p1da<p3d>dp2a<d>p3da<p1cd>:|
384:
385: (t8) @v0r*4L16@3o4@v116@k00q8
386: (t8) |:3
387: (t8) p2eb<p3_1e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
388: (t8) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da|<p3_cd>¯:|
389: (t8) p3_ga¯
390: (t8) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
391: (t8) p2da<p3_d>d¯p1a<d>p3_d<d
392: (t8) @4@v90p1o3{da<d>d+a+<d+>eb<e>f<cf>}2
393: (t8) {f+<c+f+>g<dg>g+<d+g+>a<ea>}2@5@k03p3
394: (t8) |:@v112o4|:
395: (t8) d*69r*1g-*3g*11<c8.>b8a*10b-*3b*11rf*94<e*3f*35e8der>
396: (t8) b*151<c*3c+*3d*23c8>b*67f*3f+*3g*11e4..rf&
397: (t8) f*115g*3g+*3a*23<c>bra8ge*91f*3f+*3g*23barb-&
398: (t8) b-*151<c+*3d*3d+*23>|<d4..c*10c+*3d*83r8>:|
399: (t8) <d*79¯1f*3f+*3g*11g-*85r*23_
400: (t8) e*120d8agre*156e>b<de*156agre*185r*14c*3c+*3
401: (t8) d*35cr4r*19c*3c+*3d*11c*10d-*3d*11e8.
402: (t8) d8.cr4r*19c*3c+*3d*11c*10d-*3d*11e*31c*3c+*3
403: (t8) d*35cr*43c*3c+*3d*23cdre8{d>bgg}*144ee12f12g*14
404: (t8) |:5g+*3a*23g*10:||:3a+*3b*23|a*10:|a<cc>b8a8<¯c+d*22
405: (t8) |:4d+*3e*11|d*22:|d8eg*10g+*3a*11g2r_c+rdrerf*7f*3
406: (t8) |:5f+*3g*23|f*10:|f*7g*3
407: (t8) |:3g+*3a*23|g*10:|g<c+>a+a¯<e8dc+d*22
408: (t8) |:4d+*3e*11|d*22:|d8fe|c*108r2:|c2<@5c4..r8
409: (t8) @3o4@v116@k00
410: (t8) |:
411: (t8) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
412: (t8) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da<p3_d>a¯:|
413: (t8) |:3
414: (t8) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
415: (t8) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da|<p3_cd>¯:|
416: (t8) p3_ga¯
417: (t8) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
418: (t8) p2da<p3_d>d¯p1a<d>p3_d<d
419: (t8) @4@v90p1o3{da<d>d+a+<d+>eb<e>f<cf>}2
420: (t8) {f+<c+f+>g<dg>g+<d+g+}*72_14g+_g+@5@v112@k03o4
421: (t8) a8g*10|:4g+*3a*23g*10:|
422: (t8) |:3a+*3b*23|a*10:|a<cc>b8a8<¯1c+d*22
423: (t8) |:4d+*3e*11|d*22:|d8eg*10g+*3a*11g2r_c+rdrerf*7f*3
424: (t8) |:5f+*3g*23|f*10:|f*7g*3
425: (t8) |:3g+*3a*23|g*10:|g<c+>a+a¯<e8dc+d*22
426: (t8) |:4d+*3e*11|d*22:|d8e@5bg*103 d*3e-*3e*11d8d-8deg
427: (t8) |:5f8e:||:f*10g-*3g*23:|fg8fb-8a8gfe
428: (t8) |:3d*22e-*3e*11:|dd*10e-*3e*11d8_cc*108r>b-r<c+rar
429: (t8) g8|:f*10g-*3|g*23:|g*11gf*22g-*3g*9g-*3g*11fg8
430: (t8) f*7g*3|:3a-*3a*23|g*10:|gr<¯d->b-agf_e
431: (t8) |:5d*10e-*3e*23:|ferc4..r2
432: (t8) @3o4@v117@k00
433: (t8) |:5
434: (t8) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
435: (t8) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da<p3_cd>¯:|
436: (t8) ¥21
437: (t8) |:5
438: (t8) p2eb<p3_e>e¯p1b<e>p3_eb<¯p2e>ep3_b<e¯>p1eb<p3_e>b¯
439: (t8) p2da<p3_d>d¯p1a<d>p3_da<¯p2d>dp3_a<d¯>p1da<p3_cd>¯:|
440:
441: /--------------------------------
442: / ADPCM RHYTHM
443:
444: / BASS
445: (t9) @v0r*4L16o2v9p3q4@r1
446: (t9) r1
447: (t9) |:5rd:|r4.|:6rd:|r4|:5rd:|r{ddddddddd}*60
448: (t9) |:5rd:|r8|:6rd:|r{dd}r{rd}r{dd}r{dd}
449: (t9) |:11rd:|r*168{dddd}8r8
450: (t9) |:|:|:15dr8.:|dr8d|:14dr8.:|rddr d{dd}r8:|
451: (t9) |:8dr8dr4 dr8|drdr8:|r*60
452: (t9) |:7dr8.dd|r8:|r8v10drdrdr8.v9
453: (t9) |:7dr8.dd|r8:|r8|r*96:|r2
454: (t9) |:12rddrddrd
455: (t9) |1drdd{rd}ddr:||2rddrddrd:|
456: (t9) |3drdd{rd}ddr:||4r{dd}r{rd}|:{rddd}8:|:|
457: (t9) |5drdd{rd}ddr:||6|:3d*20d*4:|{rddr}8:|
458: (t9) |7drdd{rd}ddr:||8rv10d.d.r4v9:|
459: (t9) |9drdd{rd}ddr:||10|:v7{rddd}8:|{rd}v8{dddd}{rd}rv9:|
460: (t9) |11L16drdd{rd}ddr:|r*144{dddd}8r8|:
461: (t9) |:7dr8.dd|r8:|r8drdrdr8.
462: (t9) |:7dr8.ddr8:|o4brbrbrbro2:|
463: (t9) |:5rd:|r4.|:5rd:|r4.|:6rd:|r4
464: (t9) |:5rd:|r{ddddddddd}*60
465: (t9) |:5rd:|r8|:6rd:||:r{dd}r{rd}r{dd}r{dd}||:12rd:|:|
466:
467: / SNARE
468: (t10) @v0r*4L16o4v9p3q4@r1
469: (t10) |:15dr:|d{rd}|:11dr:|d{rd}drddr8dd
470: (t10) |:16dr:||:11dr:|der2{eee}8r4er
471: (t10) |:|:
472: (t10) |:15r8er:|rddd|:14r8er:|dr*60er:|
```

▶最近「これを作ってみよう」というハード関係の記事が少なくなりました。そういう意味でもアクセラレータボードの記事を期待しています。がんばれ！浅原　英之(26)静岡県

```
473: (t10) |:31r8d|r:|{v4dv3dv4d}{dv5dv6dv8dv9dddr}4
474: (t10) |:14r8dr:|rdrdrdr{de}|:14r8dr:||re*17e.e.r*7er:|
475: (t10) rv4ddddv8dv9de
476: (t10) v8|:12dr8v9dr8v8dr
477: (t10) |1rv9dr8dr8v8d:||2|:3{v9dv7drv8d}8|r:|:|
478: (t10) |3rv9dr8dr8v8d:||4dr8dr8{ddrrrd}8:|
479: (t10) |5rv9dr8dr8v8d:||6v8r12drdrdr*14e32:|
480: (t10) |7rv9dr8dr8v8d:||8r4v9e.e.ev8:|
481: (t10) |9rv9dr8dr8v8d:||10v9drdrdrdr:|
482: (t10) |11rv9dr8v9dr8v8d:|r2v9{ddd}8r4er|:
483: (t10) |:14r8dr:|rdrdrdr{de}|:14r8dr:|rv4ddddv8dv9de:|
484: (t10) |:15dr:|d{rd}|:11dr:|d{rd}drddr8dd|:32dr:|
485:
486: / HIHAT
487: (t11) @v0r*4L32o6v9p3@q1@r1
488: (t11) |:|:8v9cv8ccc:||:8v9cv8crc:|:|
489: (t11) |:28v9cv8ccc:|v11|:6{cr}8:|{cccc}4
490: (t11) |:|:v9|:60cccc:||:3v9crv8cr:|r8:|
491: (t11) |:62v9ccv8d16:|r4
492: (t11) |:60v10cr|v8d16:|v8@r0q8d16@r1@q1r2:|L16
493: (t11) |:28v10cv6c:|v10c{cv11d}rv10c|:14v10cv6c:|
494: (t11) |:4{v10cv8cv9d}8:||:44v10cv6c:|v11|:6cr:|cccc
495: (t11) |:|:60v10cv9d:||:4v10cv6c:|:|L32
496: (t11) |:4|:8v9cv8ccc:||:8v9cv8crc:|:|
497:
498: / CRUSH
499: (t12) @v0r*4L16o6v9p3q4@r1
500: (t12) b1r*564b2b2b2b8b4@r0@q15b8r2@r1q4b2b2r*180b8
501: (t12) |:|:r*756br*744b8:|r*1512v4bv9br*756
502: (t12) @r0q8v8b32@q14b*30@r1q3 r*648|rv9b*17b.b.r*31:|r2
503: (t12) r*1140v9bbr*288v7b.b.v8b.b.v9b
504: (t12) |:br*132v8b.v9b.bb|r*168v9b:|r*252br
505: (t12) |:r*756@r0q8v8b32@q14b*30@r1q3 r*744:|
506: (t12) v9b1r*564b2|:b2b2b8b4@r0@q15b8r|r4..@r1q4:|
507:
508: / TOM
509: (t13) @v0r*4L16o4v10p3q4@r1
510: (t13) |:29br:|bv8{rfv10c}brbr
511: (t13) |:28br:|r2v9{aaa}8r.a32fr8.
512: (t13) |:|:r*732v3fff r*672v9ar8v10fr4:|
513: (t13) v8<|:7r2.ddr8:|>r*1536|r2..v6<d>r:|r*108v8ffffff
514: (t13) |:12ar8fr8v10crv9
515: (t13) |1rfr8fr8f:||2{aararrffrfrrv10ccrcv9}2:|
516: (t13) |3rfr8fr8f:||4ar8fr8{aaffv10ccv9}8:|
517: (t13) |5rfr8fr8f:||6|:3{rv9av10fcrr}8:|r8v9:|
518: (t13) |7rfr8fr8f:||8r2:|
519: (t13) |9rfr8fr8f:||10r24frfrfrfr*4:|
520: (t13) |11rfr8fr8f:|r2v9{aaa}8r.a32r4
521: (t13) |:r*1452v8ffffff:|v10w14
522: (t13) |:29br:|bv8{rfv10c}brbr
523: (t13) |:32br:|
524:
525: / CLAP
526: (t14) @v0r*4L16o5v7p3q4@r1
527: (t14) |:r4br8.r4v10bv8dr8 r4v7br8.r4br8.:|
528: (t14) |:7r4br8.:|w
529: (t14) |:r4br8.r4v10bv8dr8 r4v7br8.r4br8.:|
530: (t14) |:8r4br8.:|
531:
532: (p)
```

リスト2　スーパーハングオンのAD PCMデータ分離プログラム

```
 10 /* SUPER HANGON ADPCMデータを分解  91/11/05 by ENG
 20 /*
 30 /* このプログラムと同じディレクトリ上に、
 40 /* 「SPHSND.MOP」がある状態で実行してください。
 50 /*
 60 /* SPH_GM.PCM はゴミデータです。
 70 /*
 80 int f0,f1,i
 90 int size(9)  = {1663,1032,1752,1464, 611, 200,8717,4095,5536,1385}
100 str na(9)[2] = {"HT","MT","LT","SD","BD","GM","CC","RC","OH","CH"}
110 char pcm(20000)
120 /*
130 f0 = fopen( "SPHSND.MOP", "r" )
140 fseek( f0, 91, 0 )
150 for i = 0 to 9
160   fread( pcm, size(i), f0 )
170   f1 = fopen( "SPH_" + na(i) + ".PCM", "c" )
180   fwrite( pcm, size(i), f1 )
190   fclose( f1 )
200 next
210 fcloseall()
```

リスト3　SPLASH WAVEの音色コンフィグファイル（AD PCMデータ流用版）

```
/ -OUT RUN- SPLASH WAVE (C)SEGA  by ENG (+PCM8)
/
/ スーパーハングオンのADPCMデータを使うVer.
/ ZMUSIC付属ADPCMデータからも、いくつか必要

1=clapm1.pcm,p-1,v5
2=bass1.pcm,v30
3=cow808.pcm,v11,p3
4=casta.pcm,p-11,v17

.o2c=sph_bd.pcm,v164,m2
.o2d=.o2c,v112

.o4c=sph_lt.pcm,v120
.o4d=sph_sd.pcm,v151
.o4e=.o4d,mo2c
.o4f=sph_mt.pcm,v210
.o4a=sph_ht.pcm,v119
.o4b=tablna.pcm,v15,p-9,m3

.o5c=side1.pcm,v37,p-2,m4
.o5d=clapm1.pcm,p-1,m1,d930,v41
.o5b=shakerm1.pcm,p2,v140

.o6c=sph_ch.pcm,v82
.o6d=sph_oh.pcm,v82
.o6a=sph_cc.pcm,v188
.o6b=sph_cc.pcm,v200

.erase 1
.erase 2
.erase 3
.erase 4
```

リスト4　SPLASH WAVEの音色コンフィグファイル（Z-MUSIC標準データ版）

```
/ -OUT RUN- SPLASH WAVE (C)SEGA  by ENG (+PCM8)
/
/ ZMUSIC標準ADPCMデータ対応Ver.（ちょっと情けない）

1=clapm1.pcm,p-1,v5
2=bass1.pcm,v40
3=cow808.pcm,v13,p3
4=casta.pcm,p-11,v17

.o2c=kick33.pcm,v75,m2
.o2d=.o2c,v112

.o4c=tom7.pcm,v64,p-2
.o4d=reals.pcm,v72,p1
.o4e=.o4d,mo2c
.o4f=tom6.pcm,v62,p-2
.o4a=tom5.pcm,v58,p-1
.o4b=tablna.pcm,v15,p-9,m3

.o5c=side1.pcm,v37,p-2,m4
.o5d=clapm1.pcm,p-1,m1,d930,v41
.o5b=shakerm1.pcm,p2,v140

.o6c=hhc.pcm,v71
.o6d=ho1.pcm,v45
.o6a=crsh0.pcm,v50
.o6b=crsh0.pcm,v54

.erase 1
.erase 2
.erase 3
.erase 4
```

リスト5　SPLASH WAVEカウンタ表示

```
 1:00005722 00000000   2:000051F4 00000000   3:000051F4 00000000   4:000051F4 00000000
 5:000051F4 00000000   6:00005AC4 00000000   7:00005AC4 00000000   8:00005AC4 00000000
 9:000051C4 00000000  10:000051C4 00000000  11:000051C4 00000000  12:000051C4 00000000
13:000051C4 00000000  14:00000BA4 00000000
```

▶やはりＪリーグは熱いですね。いまのところアントラーズの独走となっていますが，まだわかりません。一度試合を見ると病みつきになりますね。　奥山　貴士(21)京都府

ASK3アクセサリプログラミング

しつこくアクセサリ

Tamura Kento 田村 健人

ASK68K ver.3.0でも辞書容量の弱さは改善されませんでした。そこで田村氏はアクセサリを使って辞書変換機能を拡張しようと，いろいろ奮闘したようですが……。さて，結果はいかに?

SASIを「さじー」と読むのはともかく，SCSIを「すかじー」と読むことを知ったときには強いショックを受けました。FEPを「ふぇっぷ」と読むことには，いまだに抵抗を感じます。

逆に，拡張子PIを「ぴーあい」と読む人には「『ぱい』って読むんだよ」と教えてあげたくなります。

ASKを「えーえすけー」と読む人は……いませんよね?

またアクセサリ

先月号にもアクセサリの例を載せましたが，説明が皆無に等しかったので，今回は丁寧にいきましょう。今回もC言語で作ります。最初の例は，

「一時的に違う辞書で変換する」

というものです。これをアクセサリで実現してみましょう。FPコールに辞書ファイルの参照/登録がありますので，素直に実現できそうです。

「違う辞書で変換する」というのは，要するにASK起動中にF8キーを押して，辞書ファイル名を入力して，変換キーを押して変換して，F8キーを押して元の辞書ファイル名を入力する，ということですね。この煩わしさを，アクセサリでなんとかしようというのが目的です。「わざわざそんなことしないで，辞書をマージすればいいじゃん」と思われるかもしれませんが，ASKには「辞書の1Mバイト制限」というものがあるので，安易に辞書を太らせることは危険なのです。

前回のフォローアップ

前回の Makefile中に，floatlib.lというライブラリが書いてありますが，これはfloatfnc.lのことです。え〜と，たしか，gcc -lでライブラリを指定すると末尾にlibという文字をくっつけて検索するので，名前を変えてしまったような記憶があります。

prefi.xなどのサンプルを実際に使ってみた方はいるでしょうか? 常駐時の占有メモリがやけに大きいことに気づきましたか? これは，Cのヒープサイズとスタックサイズを指定していないため，デフォルト値が使われているからです。本当はfplib.sの中で指定していたんですが，コメントアウトしてありますね。きっと魔がさしたんでしょう。対処法は組み込み時に，

```
A>prefi -stack:1024 -heap:0
```

などとしてください。

microEmacsでprefi.xを使うときは，CAPSを解除しておいてください。ESCのあとのキーの大文字と小文字が入れ替わってしまいます。

起動キーと終了のタイミング

このアクセサリのイメージは，「ASKに2つの変換キーがあり，片方ではこの辞書，もう片方ではあの辞書で変換する」です。変換に使うキーはXF3キーとスペースキーが一般的だと思われますので，どちらかをこのアクセサリの起動キーにしてしまってかまわないでしょう。どっちでもいいのですが，とりあえずXF3キーにします。

アクセサリを終了させるタイミングですが，アクセサリの返り値としてCACI_SUSPEND (確定した次に呼んでもらう) というものがありますので，それを使ってみます。

全体の流れは，

・XF3で起動
・元の辞書名を保存
・新しい辞書名を登録
・KEY_AGAIN CACI_SUSPENDで返って，ASKに変換してもらう
・確定したあとに呼ばれるので，元の辞書名を登録して終了

となります。KEY_AGAINを使う関係で，スペースキーとXF3キーの両方が変換キーとして使えるという前提が必要になります。このように，アクセサリの動作がASKのキーバインドに左右されるのはあまり美しくありません。本来ならアクセサリからASKのキーバインドを知る手段があるべきですが，そんなものはありません。とりあえず，アクセサリ常駐のコマンドラインで起動キーまで指定できるようにすればいいでしょうか。今回はサンプルということでそこまでやっていません。

コーディング

常駐のコマンドラインで辞書名を必要とするので，前回のkeep.cに少し手を入れなければなりませんが，これぐらい小規模のアクセサリならすぐに書けるでしょう。前述のとおりにコーディングしたプログラムの一部が図1です。

そして，このプログラムの動作は，期待どおりのものではありませんでした。XF3キーを押すと，キー入力待ちになり，なにかキーを押すとアクセサリが終了してしまうようです。どうも私はCACI_SUSPENDの意味を取り違えているようですね。だとするとCACI_SUSPENDの正しい用法は謎になってしまいますが，実際わからないのです。ごめんなさい。

ほかの方法を考えます。元の辞書名は，次に変換するときまでに戻しておけばいいのですから，

・XF3キーでは元の辞書名を保存して新しい辞書名を登録して変換
・スペースキーでは元の辞書に戻して変換

これでいいでしょう。XF3キーとスペースキーの両方でアクセサリに制御を渡してほしいので，ASKに対して2つのアクセサリを登録します。

さて，できあがったプログラムの一部が図2です。察しのよい方はもうお気づきでしょう。一部しか掲載しないということは，これでも動かなかったということです。

XF3キーで変換しても，スペースキーのときとまったく変わりありません。ところが，XF3キーを押したあとにF8キーで辞書名を確認すると，ちゃんと辞書名が変わっているのです。ASKを一度終了してもう一回起動しても辞書が変わることはありませんでした。

「プログラマーズマニュアル」を読んでも，特に注意すべきことなどは記載されていません（現時点でASK3に対応していないのだから，当然といえば当然なのかもしれない)。ただ，辞書のオープン/クローズというコールがありますので，これを試してみましょう。つまり，

・辞書を登録する前にクローズし，登録したあとにオープンする

です。

で，プログラムの一部が，図3です。動作は前のものとまったく変わりませんでした。ここまでくると，「うーん」と唸ってしまいます。あまりにも不条理なので，ASK3の処理を疑い，解析を始めました。

1) 新設されたFPコールはあるのか
2) F8キーによる辞書名変更とFPコール32番（辞書名の登録）の違いはあるのか

この2点について調べてみました。

まず1)ですが，前回紹介したアクセサリ関連のコールを除いて，「プログラマーズマニュアル」に記載されていないFPコールが1個だけありました。42番です。moveq.l #3,d0としたあと，clr.w d0して返るという，なんの副作用もなく，まったく無意味なコールです。

そして，2)についてですが，結果は「大きく違う」です。FPコール32番が，ただ辞書名のワークエリアに引数を転送するだけであるのに，F8キーによる辞書名変更では，辞書名をワークエリアにコピーしたあと（オープンやクローズといった安直なものではなく），かなり内部に立ち入った複雑な処理をしているように見られました。

図1

```
extern char paramname[];
char dicname0[FILENAME_MAX];
char dicname1[FILENAME_MAX];
int bAnother = FALSE;

short acc_main( BIT16 k ) {
    if ( !bAnother ) {                    /* XF3 が押された時 */
        int brk = BREAKCK( -1 );
        BREAKCK( 0 );
        KNJCTRLn( 41, dicname0, dicname1 );
        KNJCTRLn( 32, paramname, dicname1 );
        bAnother = TRUE;
        BREAKCK( brk );
        return CACI_SUSPEND | KEY_AGAIN;
    } else {                              /* 確定した後 */
        int brk = BREAKCK( -1 );
        BREAKCK( 0 );
        KNJCTRLn( 32, dicname0, dicname1 );
        bAnother = FALSE;
        BREAKCK( brk );
        return CACI_END | KEY_AGAIN;
    }
}
```

図2

```
short acc_main0( BIT16 k ) {      /* XF3 キーのとき */
    if ( !bAnother ) {
        int brk = BREAKCK( -1 );
        BREAKCK( 0 );
        KNJCTRLn( 41, dicname0, dicname1 );
        KNJCTRLn( 32, paramname, dicname1 );
        bAnother = TRUE;
        BREAKCK( brk );
    }
    return CACI_END | KEY_AGAIN;
}

short acc_main1( BIT16 k ) {      /* スペースキーのとき */
    if ( bAnother ) {
        int brk = BREAKCK( -1 );
        BREAKCK( 0 );
        KNJCTRLn( 32, dicname0, dicname1 );
        bAnother = FALSE;
        BREAKCK( brk );
    }
    return CACI_END | KEY_AGAIN;
}
```

図3

```
short acc_main0( BIT16 k ) {      /* XF3 キーのとき */
    if ( !bAnother ) {
        int brk = BREAKCK( -1 );
        BREAKCK( 0 );
        KNJCTRLn( 29 );                      /* クローズ */
        KNJCTRLn( 41, dicname0, dicname1 );
        KNJCTRLn( 32, paramname, dicname1 );
        KNJCTRLn( 28 );                      /* オープン */
        bAnother = TRUE;
        BREAKCK( brk );
    }
    return CACI_END | KEY_AGAIN;
}

short acc_main1( BIT16 k ) {      /* スペースキーのとき */
    if ( bAnother ) {
        int brk = BREAKCK( -1 );
        BREAKCK( 0 );
        KNJCTRLn( 29 );                      /* クローズ */
        KNJCTRLn( 32, dicname0, dicname1 );
        KNJCTRLn( 28 );                      /* オープン */
        bAnother = FALSE;
        BREAKCK( brk );
    }
    return CACI_END | KEY_AGAIN;
}
```

これでは辞書の切り替えなどできません。FPコール32番の存在意義さえありません。結局，このアクセサリの制作を諦めるしかないようです(実はこのあと，ASK3内のめぼしいルーチンに当たりをつけ，アクセサリから直接呼ぶという暴挙にも出ている。が，解析不足でことごとく失敗した)。

気を取り直して

2つ目の例は，

「透過モード」

です。ASKを終了せずに，半角英数字を入力する，です。皆さんは文書を入力しているときに半角英数字がほしくなったらどうしますか？　ASK3では入力した文字列に戻してくれるキー操作が追加されたため，それを使えばいいって？　でもこれは「入力された文字列」に対応しているので，かな入力の人は恩恵を受けられないのです(知ってましたか？)。ローマ字入力でも入力中にハナモゲラな文字列が表示されているのはイヤな人はいるでしょう。

かといって，わざわざASKを終了して，また起動するのは，時間がかかります。そうかといって，全角キーを押してローマ字キーかかなキーを押して……というのは面倒です。面倒ですが，半角英数字を単語登

変更されたFPコール

解析の結果わかった変更点をまとめておきます。

・32番，41番

ASK2までは，

```
	pea	SUBDIC
	pea	MAINDIC
	move.l	#32,-(sp)
	DOS	_KNJCTRL
	lea	12(sp),sp
```

となっていましたが，サブ辞書が廃止されたので，SUBDICにあたる引数は完全に無視されます。

```
	pea	DIC
	move.l	#32,-(sp)
	DOS	_KNJCTRL
	addq.l	#8,sp
```

でかまいません。41番でも同様です。

・42番

使い方

```
	move.l	#42,-(sp)
	DOS	_KNJCTRL
	addq.l	#4,sp
```

効用　なし

返り値　常に0

リスト1　th.c

```
  1: /*
  2:                 ASK3 アクセサリ
  3: 
  4:                   透過モード
  5: 
  6:                                   by けんと
  7: */
  8: 
  9: #include          <stdio.h>
 10: #include          <doslib.h>
 11: #include          <iocslib.h>
 12: 
 13: #include          <aci.h>
 14: #include          <askkey.h>
 15: int KNJCTRLn( int, ... );
 16: enum { FALSE, TRUE };
 17: 
 18: 
 19: short acc_main0( BIT16 k );
 20: short acc_main1( BIT16 k );
 21: 
 22: unsigned char dbuf[80];
 23: MEAN kbuf[80];
 24: MEAN mbuf[20];
 25: int iAccHandle0;
 26: int iAccHandle1;
 27: 
 28: const ACC_DEF accdef0 = {
 29:         KS_EDITING | KS_EDIT0 | KS_SELECT,
 30:         SHIFT_ON | NOT_ASCII | ZENKAKU_OUT,
 31:         acc_main0,
 32:         {
 33:                 dbuf, kbuf, mbuf
 34:         }
 35: };
 36: const ACC_DEF accdef1 = {
 37:         KS_EDITING | KS_EDIT0 | KS_SELECT,
 38:         SHIFT_ON | NOT_ASCII | ZENKAKU_IN,
 39:         acc_main1,
 40:         {
 41:                 dbuf, kbuf, mbuf
 42:         }
 43: };
 44: 
 45: int sk = 0x0230;
 46: 
 47: 
 48: 
 49: short acc_main0( BIT16 k ) {
 50:         int brk = BREAKCK( -1 );
 51: 
 52:         BREAKCK( 0 );
 53:         sk = K_SFTSNS();
 54:         LEDMOD( 0, FALSE );                     /* かな */
 55:         LEDMOD( 1, FALSE );                     /* ろーま */
 56:         BREAKCK( brk );
 57: 
 58:         return CACI_END;
 59: }
 60: short acc_main1( BIT16 k ) {
 61:         int brk = BREAKCK( -1 );
 62: 
 63:         BREAKCK( 0 );
 64:         if ( sk & 0x010 )  LEDMOD( 0, TRUE ); /* かな */
 65:         if ( sk & 0x020 )  LEDMOD( 1, TRUE ); /* ろーま */
 66:         BREAKCK( brk );
 67: 
 68:         return CACI_END;
 69: }
 70: 
 71: 
 72: 
 73: int iOpenProgram( void ) {
 74:         if ( KNJCTRLn( 50 ) < 300 ) {
 75:                 printf( "ASK.SYS v3.00 以上が登録されていません。¥n");
 76:                 return 1;
 77:         }
 78:                                                 /* アクセサリを登録 */
 79:         if ( (iAccHandle0=KNJCTRLn( 60, &accdef0 )) < 0 ) {
 80:                 return 1;
 81:         }
 82:         if ( (iAccHandle1=KNJCTRLn( 60, &accdef1 )) < 0 ) {
 83:                 KNJCTRLn( 61, iAccHandle0 );
 84:                 return 1;
 85:         }
 86:         printf( "¥n透過モード for ASK3  by けんと¥n"
 87:                 "SHIFT+全角 で動作します。¥n" );
 88:         return 0;
 89: }
 90: 
 91: 
 92: int iCloseProgram( void ) {
 93:                                                 /* アクセサリを削除 */
 94:         if ( KNJCTRLn( 61, iAccHandle0 ) < 0 ) {
 95:                 return 1;
 96:         }
 97:         if ( KNJCTRLn( 61, iAccHandle1 ) < 0 ) {
 98:                 return 1;
 99:         }
100:         return 0;
101: }
```

録したいときはやらねばならないのです。そこで,「全角キーが消えるときに,一緒にローマ字キーとかなキーも消えてくれたら便利ではないか」と考えます。また逆に「全角キーが点灯するときに以前の状態に戻ってくれると幸せ」と。この間,打ったキーがそのまま入力されるので,「透過モード」と名づけます。

まさか全角キー本来の機能を殺してしまうわけにはいかないので,透過モードとの切り替えを「SHIFT＋全角」で行います。(本来の全角キーを「SHIFT＋全角」にして透過モードを「全角」にするのもいいかもしれません)

コーディングに入ろうとして,ASKKEY.H を見ると,全角キーは消灯時と点灯時で別になっています。よって,2つのアクセサリを登録することになります。

はっきりいって,このアクセサリも大したものではありません。全角キーが消えるときにLEDの状態を保存して,かなとローマを消す,全角キーが点灯するときに元に戻す,これだけですから。

前回のkeep.c,ACI.H,ASKKEY.H,今回のth.c,knjctrln.sを入力して,Makefileを参考にコンパイルしてください。homy氏による移植のGNU makeを使うときは,環境変数MAKE_SHELLが参照されるので気をつけてください。

アクセサリの可能性

前回と今回で作ったアクセサリを改めて見てみると,4つ中3つがキー入力の補助だといえます。

この記事を書く前に,どのようなアクセサリを作るかということで,ASKに対してどのような不満があり,どのような拡張をしたいかを考えて,多数の候補が挙がりました。その候補のうち,半分ほどは,現段階のアクセサリでは実現不可能でした。どれも,変換中の文字列を置き換えられさえすれば実現できるものです。

アクセサリから「変換」という処理ができないことがどれだけ自由度を下げているか,痛感しています。辞書を切り替えることができれば,思いどおりの変換処理をすることもできましたが,辞書の切り替えはできませんでした。

日本語入力とは関係のないアクセサリならいくらでもできるでしょう。ASKが起動できるところならアクセサリも起動できるわけですから,いつでも起動できるためのプラットホームとして見るのも面白いかもしれません。

データベースとしてのアクセサリ

今回のASK3辞書切り換え計画は見事に頓座してしまいましたが,アクセサリから扱うものをASK68Kの辞書というかたちではなく,たとえば独自形式の辞書で持っておけば普通のファイルアクセスプログラムだけで簡単に使用することができます。

そうなるとこれはデータベースアクセスの一例のようなものです。ファイルを開いてキー文字例に対応するデータ群を拾ってきて,選択候補にすることになります。文字例の変換位置が取ってこれないので,アクセサリ内で文字例を切り出してやることになります。このようにするとASK2互換のキー操作を実現できるかもしれません。

もっとも,こういったものを進めていくというのは,日本語入力FEPのカーネルを自作していくのとなんら変わりないということになってしまうのですが……。

＊　　＊　　＊

普通はもっと普通のデータベース(変ないい方だな)をアクセスするのが筋というものでしょう。〜のデータベースにアクセスするには○○キー,〜のデータベースなら××キー……といった具合に使い分けることで,音楽用品や化学用語,医学用語などの特殊な用語でも的確にアクセス可能となります。　こういったデータベースでは扱うデータが違うだけで,やっていることはほとんど変りありません。まず,基本的なデータベースフォーマットというのを決めてしまえば,あとは誰にでも専用データベースが作れるようになるでしょう。

リスト2 knjctrln.s

```
 1: *           FPコール呼びだし
 2: *
 3: *                              by けんと
 4: *
 5: *   この関数は d0/a0(?)/a1 を破壊します。
 6:
 7:
 8:                 .include        doscall.mac
 9:
10: *               .global _HEAP_SIZE,_STACK_SIZE
11: *_HEAP_SIZE     equ     0
12: *_STACK_SIZE    equ     1024
13:
14:
15: _KNJCTRLn::
16:                 move.l  (sp)+,a1
17:                 DOS     _KNJCTRL
18:                 jmp     (a1)
19:
20:                 .end
21:
```

リスト3

```
 1: CC       =   gcc
 2: CFLAGS   =   -Wall -O -fomit-frame-pointer -fstrength-reduce ¥
 3:               -fcombine-regs
 4: AS       =   has
 5: ASFLAGS  =   -w2
 6: LD       =   hlk
 7: LDFLAGS  =   -x -v -d _STACK_SIZE=400 -d _HEAP_SIZE=0
 8: LIBS     =   clib.l floatfnc.l doslib.l iocslib.l
 9:
10:
11: ALL: through.x
12:
13: through.x: th.x
14:              mv $< $@
15:
16: %.x: %.o
17:              $(LD) $^ $(LDFLAGS) -l $(LIBS)
18:
19: th.x: knjctrln.o keep.o
20:
21: %.o: %.s
22:              $(AS) $(ASFLAGS) $< -o $@
23:              chmod +h $@
24:
25: %.o: %.c
26:              $(CC) $(CFLAGS) -c $< -o $@
27:              chmod +h $@
```

新製品紹介

Easydraw SX-68K

Tan Akihiko 丹 明彦

X68000用　3.5/5"2HD版　価格未定／シャープ　☎03(3260)1161

待望のドローツールがSX-WINDOWに，いやX68000シリーズに登場した。サクサクと簡単な説明図の描けるドローツールを，マルチタスクの環境下で利用することができるようになったのだ。

▶ ドローツールとは何か ◀

グラフィックツールで一般的なのはペイントツールと呼ばれる，画面の物理的なドット(ピクセル)を編集するソフトだろう。これに対し，ドローツールとは図面を論理的な部品(ドローオブジェクト)の単位で組み立てていくもの。長方形や円，多角形などの図形を重ね合わせながら図面を作り上げていく。点を打つのが基本であるペイントツールとは，根本的に思想が異なるのだ。

X68000用に発売されていたグラフィックツールのほとんどは，「Z'sSTAFF PRO-68K」「MATIER」を代表とするペイントツールである。唯一のドローツールである「CANVAS PRO-68K」は表現力こそ高かったが，速度や手軽さという点でとうてい満足のいくものとはいえなかった(X68030ならそこそこ使える速度で動作する)。

SX-WINDOWが世に出て，早3年。Macintoshのススんだ DTP環境(といっても「マックドロー」のような手軽なものが好みだが)を横目に見ながら長い間，本当に長い間，待ち望んでいたドローツールが使えるのである。これが喜ばずにいられようか。

▶ ドローのいいところ ◀

ドローツールのメリットをいくつか挙げよう。もちろん，ドローツールとペイントツールはどちらが優れているというわけではなく，そもそも用途が異なるものであると思っていい。端的にいえば，ペイントツールは絵を描くもので，ドローツールは図を描くものである。

○拡大縮小自由自在

ドローオブジェクトは論理的な部品である。たとえば，拡大してもドットが粗くなったりしないし，縮小してもドットが潰れたりしない。線はあくまで線，多角形はあくまで多角形である。

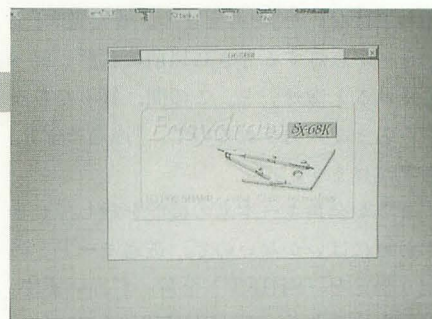
○印刷が美しい

拡大縮小が自由に行えることと関連するが，プリンタのドットを目一杯使った美しい印刷ができる。曲線はあくまで美しく，文字にベクトルフォントを用いれば，そこはDTPへの第一歩なのである。

○切り張りが楽

ドローオブジェクトが論理的な部品であることと関連するが，ドローオブジェクトを移動したりコピーしたりといったことが極めて容易に行える。文字どおり，図形の「編集」という作業を行うための強力なツールなのである。

○描いた図の再利用が容易

ドローオブジェクトは部品の集まりなので，いつでも好きな部分を取り外して，ほかの図に持っていける。図の使い回しは，ドローツールの最も得意とするところであろう。

▶ 機能概要 ◀

1)　図形の作成。オブジェクトは直線，曲線，長方形，円，多角形などが使える。基本中の基本。

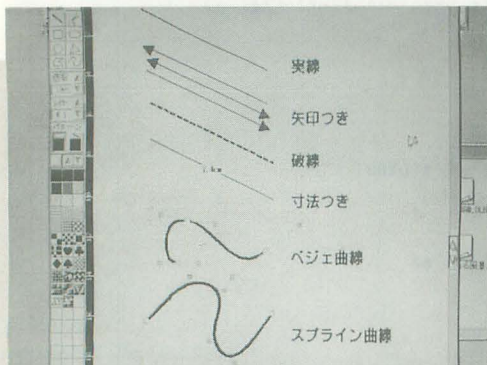
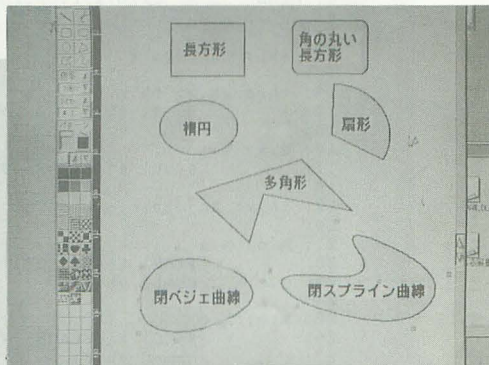
2)　ドローオブジェクトのリサイズ(拡大/縮小)，移動および変形，回転。ドローツールの真骨頂。

3)　文字の作成，自由な文字種，および文字サイズの設定。文字種はROMフォントにもベクトルフォントにも対応。私は書体倶楽部の「角ゴシック(中)」が好みだ。文字サイズはドットだけでなく級，ポイント，ミリメートルの中から選べる。

4)　ドローオブジェクトのグループ化。ひとかたまりのドローオブジェクトをグループとして，あたかもひとつのオブジェクトのように振る舞わせることができる。

5)　多彩な装飾指定。多角形などの面にスクリーントーン上の模様を張り付けたり，線の太さや実線/破線が設定できる。

6)　「CANVAS PRO-68K」のドローデータが読み込める。過去の資産を持っている人にはおいしい。「ドローグラフィックライブラリ vol.1/2」というパッケージも，シャープから発売されているし。

7)　レーザープリンタドライバ装備。エプソンのESC/Pageやキヤノンの LIPSIII，ポストスクリプトなどに対応する。レーザープリンタは美しさと速さを兼ね備えたプリンタで，まっとうなDTPの必需品といえる。

ドローツールはパーツが命

図版をつくるのにはもってこい

印象，その他

○操作性

複雑な図だと10MHzのマシンでは少しつらいが，サンプルのような図なら楽に描ける。X68030だとまったく問題ない。

○印刷

私はBJ-10vユーザーである。多少印刷に時間のかかることを除けば，おおむね満足。360dpiの解像度はだてじゃないな。

印刷時だけでなく，編集時もちゃんとA4版の紙(というか用紙設定で決めた紙)を意識した作りになっていることもうれしい。ドットを意識せずに使えるというのはいいものだ。どうして，いままではこれができなかったのか不思議だ。印刷は正しくマルチタスクで行っているようだ。

レーザープリンタ関係についてはまだ入っていなかったようで試してみることはできなかった。完成がとても楽しみである。

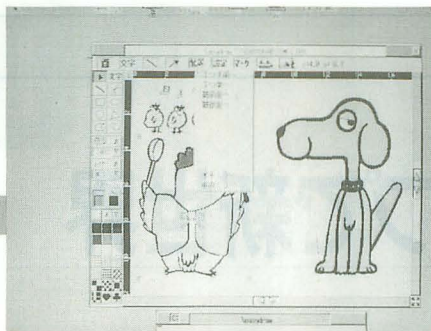
○再利用性

現時点では，Easydraw.Xのドローオブジェクトをカットして，シャーペン.Xの文書にペーストすることができない。イメージペーストなら可能なのであるが，それではドローオブジェクトの拡大縮小自由自在という特徴が生きてこない。これに限らず，シャーペン.Xにはデバイス非依存に関する考慮が欠けているような気がする(ドットにかなりとらわれている)。

これはシャーペン.Xの改良か，「EG Word SX-68K」の登場を待つことになるだろう。いや，そもそも，ペイントオブジェクトと同様，ドローオブジェクトのカット&ペーストがシステムレベルで自由にできるようにしておく必要がある。

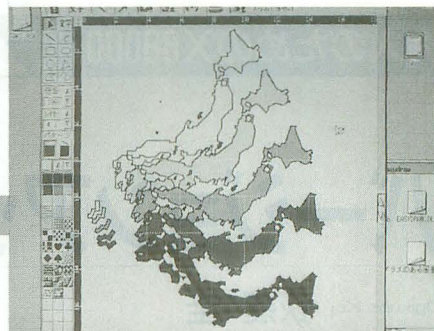
○文字入力

インライン変換はあったほうが便利，というか，早急にシステムレベルでインライン変換のメカニズムを確立しておいたほうがあとあと困らないと思う。

○あくまで軽くて小回りの利くツール

サンプルでついてくるデータ

サンプルのドローデータや利用できる英数字の書体を見ると，ちょっと不安である。「CANVAS PRO-68K」のデータをコンバートしたのだろうが，このまま「ポップアートツール」への道を歩まないことを望む。表現力が高いけど操作が重いものよりも，サクサク描けるもののほうがいい。

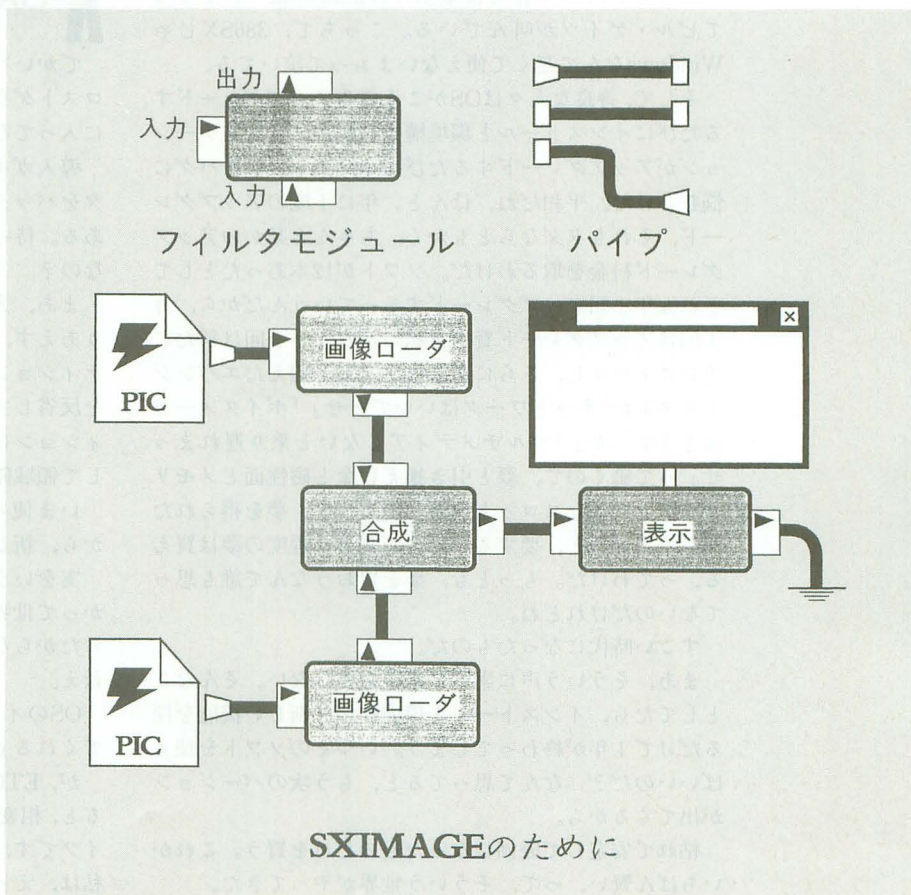
○実用性を決めるのはフォント

これは「Easydraw SX-68K」のせいではないが，SX-WINDOWにもいいかげんにまともな英数字フォントの拡充が望まれる。少なくとも明朝体，ゴシック体とバランスのとれるような書体は用意しないと使い物にならない。マルチフォントエディタといいながら，英数字には装飾文字しか用意していないのは気になる。どうにか我慢できるのは「Trad」だけというのではあまりに情けない。てっとりばやいのはベクトルフォントの仕様を公開してしまうことである。

回転させても当然きれい

終わりに

今回はβバージョンということで，細かいバグも多かったが，気をつけて使えば問題なく図は描けた。とりあえずほしい機能はおおむね揃っているし，完成した暁には必携のツールとなることであろう。

職業柄，手軽に立ち上げられてサクサク描けるドローツールがほしかったのだ。

出力例 (BJ-10v)

バージョンアップで新世界

Ogikubo Kei 荻窪 圭

めまぐるしく変化するパソコン界。そのなかで，よその騒ぎもどこ吹く風と独自の世界を作っているX68000ですが，ついに新しい波の出現です。これからどう変わっていくのか，ちょっとのぞいてみましょうか。

もう，いったい何を急いでいるのか，って感じ。世紀末に向かってどんどん加速していき，1999年に全部爆発してはじけ飛んじゃうんじゃないかって思ってしまうくらいだ。何の話，ってパソコン世界の，さらにはデジタルデータ処理世界の話。

インテルよ，そんなにCPUの開発を急いでどこへ行く。マイクロソフトよ，そんなに次々とビジョンばかりでかくしてどこへ行く。もう，ややこしくて困る。

買い替えろ買い替えろ買い替えろ。買い足せ買い足せ買い足せ。あおうるさい。

構想ばかりが等比級数的に膨らんでいき，現実は遅々としてははとして，様子見ばかり。先端と末端の差は広がるばかりで，確かに10年前といまではあまりに違う環境となり，誰にも予測不可能な世界が構築されたわけだが，人間なんてそんなに進歩しないもの，一太郎Ver.3レベルの感性しかもってない者に何を説こうというのか。

あっちでFAXや留守番電話にWindows載っけようってビル・ゲイツが叫んでいる。こっちで，386SXじゃWindowsなんて遅くて使えないよぉって泣いてる。

そして，善良な人々はOSがころころアップグレードするたびにインストールと環境構築に悩み，アプリケーションがアップグレードするたびにインストールとバグに悩むわけだ。平和だね，ほんと。年に1度のアップグレード，それもタダならともかく，ちゃんと某(なにがし)かのアップグレード料金を取るわけだ。ソフトが12本あったとしてどれも年1回アップグレードするっていうんだから，月1回はアップグレード費用を払い込み，月1回は新たにインストールし，さらに新しいソフトを抱えたエバンジェリストが「ネットワークはいいでっせ」「ボイスメールはどうでっか」「マルチメディアしないと乗り遅れまっせ」って囁くので，夢と引き換えに金と磁性面とメモリチップと貴重なスロットを差し出す。で，夢を得られたか，っていうと，要するに，金で買える程度の夢は買える，ってわけだ。もっとも，夢を買おうなんて誰も思ってないのだけれどね。

すごい時代になったものだ。

まあ，そういう声に素直に乗る必要はない。そんなことしてたら，インストールと環境整備と新しい機能を探るだけで1年が終わってしまう。いつそのソフトを使えばいいのだ？　なんて思ってると，もう次のバージョンが出てくるから。

枯れて安定して廉価になったものだけを買う。これがいちばん賢い，って，そういう世界がやってきた。

私なんかユーザーだからまだいいけど，渦中の開発者は大変だね。やっと新しいOS用のソフトが完成したと思ったら，もう，次のバージョンのOSがβ版になってたりして，対応しなけりゃならない。

OSを作っている人ももっと大変か。ハード作っている人も大変だ。

その先端を競っているIBMとアップルとマイクロソフトと，それからノベルも大変だ。

日本のユーザーははやく日本語化しろ，って叫ぶし。独自路線のメーカーも大変。アップルやマイクロソフトほどマンパワーをつぎこむわけにもいかないから。我が道を行くAMIGAはいいとしても（詳しくはよく知らんが），X68000なんかはどうするんでしょう，なんていってたら出ましたですよ，Human68k ver3.0とSX-WINDOW ver3.0が。

SX-WINDOW&Human68k

でかいパッケージには（それでも，ver2.0に比べると，コストダウンのためか，箱も紙になったし，FDもケースに入ってなかったけど），FD6枚とマニュアルが数冊。

導入ガイドを見ると，なんと，ハードディスクのデータをバックアップしてフォーマットしろ，なんて書いてある。待ってくれよぉ。ほんとに再フォーマットが必要なの？

まあ，大丈夫やろ，ってことで，それはやんない。とりあえず，たった40MバイトしかないHDDを6つにパーティションを区切る，なんて無謀なことをしていた自分を反省して，後ろ3つを統合してひとつの大きなパーティションにする。3つを領域解放して，ひとつのものとして領域確保するわけだ。

いま使っているHuman68k ver2.0をつぶすのは怖いから，新しい領域にインストールするわけ。

実をいうと，私，最近ずっとMacintoshとかPC/ATとかって世界にどっぷり浸かっていたから（98は捨てちゃったからないけど），身体が楽をするのに慣れてしまってねえ。

OSのインストールってのはインストーラが全部やってくれるものだと思っていた。

が，ETCディレクトリに入っているインストーラを見ると，相変わらずの相変わらず。ハードディスクはCドライブです，なんて決め打ちしてやがる。あのねあのね。私は，えっと，Fドライブにインストールしたいのよ。

しょうがないから，手作業でやる。といっても，COPYALLコマンドでもって，全部ハードディスクへコピーするだけだけど。

ついでにFD５枚にも及ぶSX-WINDOWを同様に，COPYALLコマンドを使ってコピーする。SX-WINDOWに至っては，インストーラもついてない。マニュアルに従ってコピーすると，アクセサリ類なんかがみなルートディレクトリに入ってしまう。これは美しくない。アプリケーションディスクの内容は，ハードディスクに専用のディレクトリを作って収める。

こっからが大変だ。Human68kにはいろいろと知らないコマンドが増えているし，いままで使っていた至便なユーティリティがそのまま使えるかどうかもわからない。

まず，CONFIG.SYSだ。これはver2.0の環境で使っていたヤツを睨みながら，整える。

次，AUTOEXEC.BAT。

最も重要なRENDRV.Xの動作を確かめる。以前，泉大介氏が作ってくれたプログラムで，ドライブ名を変更するものだ。なにせ私，常に，FDDはAドライブとBドライブ，HDDはCドライブからはじまってくれないと頭がパニックする人間だから，RENDRV.Xが動いてくれないと困る。

動いてくれた。

とりあえず，COMMAND.Xの環境で起動するように組む。

それにしても，見慣れないものがたくさんあるなあ。困ったものだ。

HDTVについて

前回（っていうと６月号になってしまうのが我ながら情けないわけだが），ちょろっとHDTVの話を書いた。で，書きながらいろいろと不安があったりして調べてみると，こいつがまたややこしい。

HDTVってのは，High Definition Television。訳すと高品位テレビ。現在のテレビでは解像度は低いわ色は悪いわでどうしようもない，ってんで，次世代のテレビの規格を作っているわけだ。いまのように，NTSC，PAL，SECAMなんて３つに分かれてしまうと，PAL仕様のAMIGAが日本で使えない，みたいなことが起きる。いや，もっと問題なのは，ビデオデッキが違っちゃうし，ビデオソフトも違っちゃうってことなのだ。

で，うまくいってるか，っていうと，いってるはずがない。ちゃんと，日本の提唱する規格（いわずもがな，ハイビジョンね）と，欧州の規格（HD-MAC）と，アメリカの規格ができてきた。これ，日本のハイビジョンが先に動いたのだが，アメリカが「日本に規格を押さえられたらかなわん」ってなもんで反対した，ってことらしい。で，ハイビジョンがデジタル＋アナログであるのに対し，フルデジタルで対抗しよう，と。だからといって，アメリカ側の規格がきちんとできているかっていうと，いろんなメーカーがいろんなところと手を組んで，４つほど方式ができていて，やっと，「どれか１本に絞りましょうね」って合意がとれた段階にきたところのようだ。

日本だけハイビジョンってことになるのかなあ。なるのだろうなあ。実用化に向けていちばん近いのがハイビジョンで，フルデジタルの壮大な計画は次々世代テレビってことになりそうな気もする。

うーん。でも，考えてみれば，HDTVの映像をデジタルで全部処理したら，凄いデータ量だよね。

アメリカの場合，スーパーハイウェイとかいう，アメリカ中に光ファイバーの通信網を張り巡らして高速デジタル回線何でもあり状態を作ろうという壮大な計画があるから，これが実現すれば，いろんなプロジェクトが一気に進むんだろうな。日本のB-ISDNはどうなったのだろうか。

ようわからん。ちなみに，参考文献は『日経エレクトロニクス』（日経BP社）と『HDTV最前線』（朝日新聞社）。

では，現行のNTSCはそんなにひどいのか，っていうと，スタジオ品質（つまり，伝送する前の品質）と受信品質のあいだにはえらい違いがあって，受信品質といっても，受像機の性能による違いもあって……。うちのテレビは情けないことに，いまだに15インチのあのX68000用ディスプレイテレビというていたらくで，５年前の代物だから，かなり画質的にもガタがきているわけで問題外だけども。あるとき，私の友人が（業界大手の某レコード会社の映像部門にいるのだが）LDを１枚持ってきて，せっかくだからきれいな映像で見ようと苦労して，LDプレイヤーのS端子からMacintoshに突っ込んであるビデオキャプチャ兼フルカラーボードにつなぎ，そこからダイレクトにVGAディスプレイにフルカラーで出力してみた。フルスクリーンデジタル表示なのであるが，全部ハードでやってるから，ちゃんと動くのだ。これがまた，色が全然違うのである。ほんとにきれい。

その友人曰く「これ，マスモニの色にすっごく近い」。マスモニってのは，どうやらマスターモニタのことらしい。つまり，スタジオ品質に近い映像が得られたわけだ。これを，NTSC信号にしてテレビに入れてやると，悲しくなるほど汚い。ああ，こういうものなのだなと。

そいつにHDTVのことを聞いたら，「何になっても同じよ。どうせ35mmフィルムには勝てないんだから。重要なものは全部35mmフィルムで残しておけばいい」とぬかしやがった。だから業界人はきらいだ。

それはともかく，デジタルビデオで高画質になったからとて，テレビ受像機が，つまりモニタが悪ければどうしようもないし，モニタがよくても，ビデオデッキが悪ければどうしようもないし，いざ，HDTVの時代がきたら，総とっかえだなあ，と。となると，総とっかえする気になるだけのものでないと，普及には時間がかかるなあ，と。

そもそも，そんなワイド画面のテレビなんて，逆立ちしたって置くところがない。

なんやようわからんコマンドが増えた

Human68k ver2.0で使っていたデバイスドライバなんかが全部そのまま使えるか，っていうと，うーん，ち

ょっと自信がない。それに，不要になったドライバもあれば，フリーウェアに頼らなくてもよくなったドライバもある。

いちばんうれしいのは，なんといっても，FDDEVICE.Xってデバイスドライバ。いままではむりやりフリーウェアを使って読み書きしていた5インチのIBMフォーマット2HD(2HCなんて言い方，死語だと思う)。これがなんの問題も気遣いもなく，そのまま読み書きできる。これはうれしい。とはいえ，5インチドライブを積んだPC/AT互換機を持っている人がそういるとは思えんし，普通は3.5インチドライブユーザーが喜ぶんだろうな。

対応するドライブさえつなげば，何でもあり。マニュアルを信じると，1.4Mバイトの2HDはダメっぽいけど，FDDEVICE組み込み時のメッセージではちゃんと「2HD(1.44MB)の読み書きが」って書いてある。真相はどうなんでしょう。たぶんマニュアルの間違いだろう。

仮想ドライブをこしらえるのではなく，全部同じドライブで自動的に認識してくれるところがうれしい。

次いで面白いのが，CONFIGED.X。CONFIG.SYSにEXCONFIGってコマンドができて，そのパラメータがCONFIGED.X。CONFIG.SYS専用のエディタで，起動時にSHIFTキー（他のキーに変更可）を押していると，デバイスドライバなんかを読む前に，こいつが立ち上がる（写真1）。

いやはや，面白い。これで，CONFIG.SYSの任意行をON/OFFしたり，書き換えたりできるのだ。いろんな環境変更の実験とか，メモリの少ない人なんかは用途によってCONFIG.SYSのパターンを変える，なんていう芸当ができて大重宝である。

FASTIO/FASTOPEN/FASTSEEKなんてのはいまさら私がコメントすることもあるまい。

CONFIG.SYSとAUTOEXEC.BATがいままで使っていたものの半分以下の長さになった。ただ単に，無駄なものをいっぱい突っ込んでいたせいだ。

5年前のマシンだもんな

ああ，いい加減，どうしようもなくなってきた。って，5年以上も前のマシンに向かってそんなことをいってたら，98ユーザーに笑われる。5年前のPC-9801なんて，いま持っていても博物館にも置いてもらえないし，かといって役に立つわけでもない。5年間といえば，減価償却が完了するのに十分な年月。

時代は変わったなぁと思うのは，いつのまにか640×480ドットで16ビットカラーとか24ビットカラー(フルカラー)が当たり前になったこと。X68000が出た当時は考えられなかったからね。

当時は，おそらく，512KバイトのVRAMで16ビットカラーを出すには，ってことで，512×512ドットってのが出てきたのだと思う。16色モードのときは1024×1024ドットなんて無謀さだしね。

640×480ドットってのは，ドットの縦横比が1：1になって，さらに，NTSCに変換して出力する（あるいは，その逆）には都合がいい値なのだが，これで16ビットカラーにしようと思うと，600Kバイト必要になる。すると，メモリを256Kバイト単位で用意する場合，768Kバイト必要で，168Kバイト余ってしまう。無駄なわけだ。

640×480ドットでもNTSCにするとオーバースキャンするけど，それはしょうがない。

IBMのVGAなんて，640×480ドットの16色だけど，これ，150Kバイトあれば事足りる。でも，メモリの都合で，256Kバイト載っている。こういう感覚なのだ。

どう考えても，5年前にビデオRAMを1Mバイト搭載していた，ってこと自体，凄かったし，X68000のメモリの使い方を見ると，ドットの縦横比が1：1でなおかつ16ビットカラーを，なんて無謀だというのはわかっているけれども，せっかくスーパーインポーズまでつけたのに，せっかくテレビと相性がいいっていうX1のあとを継いだのに，と思わざるをえないけど。

秀丸エディタについて

Windows用のシェアウェアエディタに秀丸エディタってのがある。こいつが非常に高速で使いやすい。

なによりうれしいのが，右ボタンのポップアップメニュー。範囲指定して右ボタンを押してやると，コピー&ペースト系のほかに，UPPER CASE/LOWER CASE，半角全角変換，ひらがなカタカナ変換などなど，お得な作業がその場でできるのだ。うーん，SX-WINDOW，ってわけで，非常に気に入っていて，キーカスタマイズを一部microEMACS風にして遊んでいる。

SX-WINDOW ver3.0

バージョン3.0といえば，某Windowsみたいだなあ，次はSX-WINDOW NTか，なんて馬鹿なことをいってる場合ではないですね。

SX-WINDOWの目玉といえば，グラフィックウィンドウ。これは無謀というかなんというか，1024×1024ドットの仮想画面中に，最大512×512ドットの「覗き穴を開けてしまおう」って技だ。普段はテキスト画面上にマスクがかかっていてグレーだけれども，いざ，グラフィックを表示しようとすると，そこにウィンドウができ，グラフィック画面への覗き穴が開くのだ。で，穴の向こうに画像がロードされている，って寸法。ドットの縦横比が違うから，X68000で作った画像をここに表示させるとかなり縦長になってしまうけど，そのへんは調節できるようになっている。面白いけど，遅いし（遅いのは私のマシンか），鑑賞用だね。

まあ，一度ここに表示したやつをPAT4フォーマットでセーブして，でもって，SX-WINDOWの背景に使う，とかすれば面白い。

このグラフィックウィンドウってけっこう邪魔だから，普段は右下の隅に追いやってある。仮想画面パワー炸裂だ。

仮想画面＋グラフィックウィンドウでスプライト以外の全VRAMを使い切ったってことだね，SX-WINDOWも。なかなか，究極にまでいっている。

小さい窓で絵を動かすことについて

IVM.Xってのを起動し，CGビジョン.Xってのを使うと，グラフィックウィンドウ内で動画を動かすことができる。

小さい窓で動画を動かす，ってのは，動画を動かす，ってことに意義があるのではなくて，

1) アニメーションから自然画像（自然画像ってのも変な言い方だけど，ほかに思いつかなかった）まで，あらゆる種類の画像を分け隔てなく扱う

2) 動画のフォーマットを統一し，なおかつ時間の管理を行う

3) ハードディスクやCD-ROMからロードしつつ再生するから長さの制限がない（もちろん，ハードディスクの容量によるが）

4) デジタルビデオ時代を見据える，っていうような意味があるもの。システムがライブラリでフォローすることによって，システムの標準形式として，アプリケーションから動画ファイルへのアクセスが容易になる，ってのもある

この手の動画フォーマットといえば，アップルのQuickTime，QuickTime for Windowsやマイクロソフトのに Video for Windowsが有名で，あと，富士通のTOWNS用ライブビジョンなんかがある。

少なくとも，QuickTimeとVideo for Windowsはデジタルビデオ時代を見据え，いまでも，金さえ積めばフルスクリーンフルフレームフルカラー（つまり，640×480ドットで秒30フレームで1600万色）のデジタルビデオを処理できる。しかし，まあ，JPEG系の圧縮がかかってしまうから，画質はそんなによくはない。

専用のアクセラレータなしで，いまのCPUでは320×240ドットで秒15フレーム，色数は16ビットカラー程度が限界ではないかと思う。もちろんこれは自然画像の場合で，CGアニメーションの場合は，もっと軽いから(圧縮がきくから)，秒30フレームくらいいくだろうと思う。

さらに，ハードディスクからロードしつつ再生するわけで，一度ムービーデータをロードしてから再生してやるのなら，もっとフレームレートを上げることはできるが，その代わり，データ量は多くなるし，そのデータを再生できるかどうかは搭載メモリに依存してしまう。

かといって，ハードディスクから再生すると，読みながら描くわけで，ハードディスクからのデータ転送速度が問題になるし，データ転送速度を稼ぐにはデータを圧縮して小さくしなければならず，圧縮したならしたで，それを展開するのにCPUタイムを食われる。

いまフルスクリーンフルフレームフルカラーをやるには，5Mバイト/secクラスの高速なハードディスクとそのインタフェイス，JPEGデータの展開を行う高速なハードウェア，68040クラスの高速なマイクロプロセッサが不可欠だ。

CGビジョン.Xがどんなフォーマットでデータを持っているかはわかんないけど，そんなに圧縮効率のいいものではないようだから，あくまでもCGアニメーションの

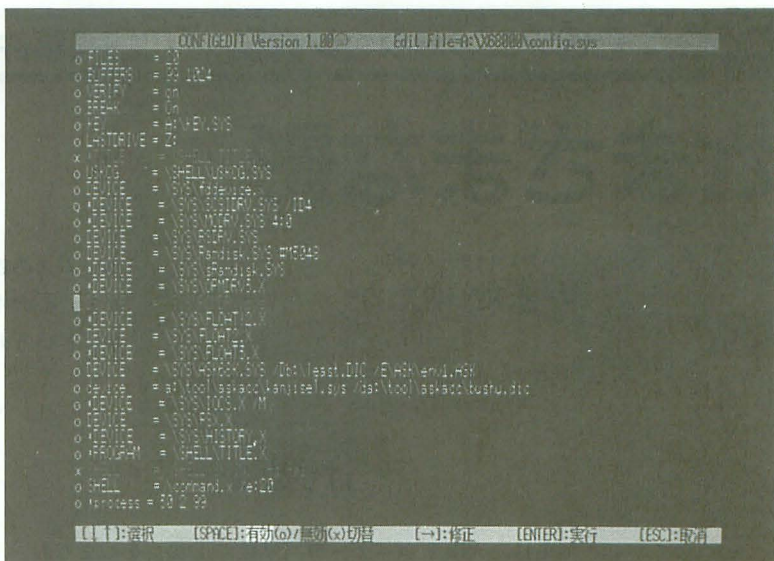



写真1　CONFIG.SYSの書き換えが楽になった

ビューワーとして考える程度のものだろう。

小さい窓で絵を動かすってシステムは，将来はちゃんとしたデジタルビデオするぞ，っていう目標への第一歩であると同時に，絵だろうが動画だろうが，デジタル化されたデータは何でもかんでもオブジェクト化して，テキストやらなんやらと同じレベルで扱おう，って野望をもっている。これはとりもなおさず，パソコンがただの制御装置からメディアへと進むための野望だ。パソコン内で完結し，パソコン内でしか扱えない作品を作ること，といってもいい。ひとつの作品はさまざまなオブジェクトの集合体となり，そこに組み込まれるオブジェクトとして重要なのはフルスクリーンで動くデジタルビデオではなく，サイズは小さくてもより制限のない自由なデータだ。

そんな都合のいいシステムがいつ完成するかはわからないけど，それに15Mbpsなんていう高速デジタル回線なんかが加われば（まあ21世紀のことだろうけど）かなり面白いことになるのではないか。

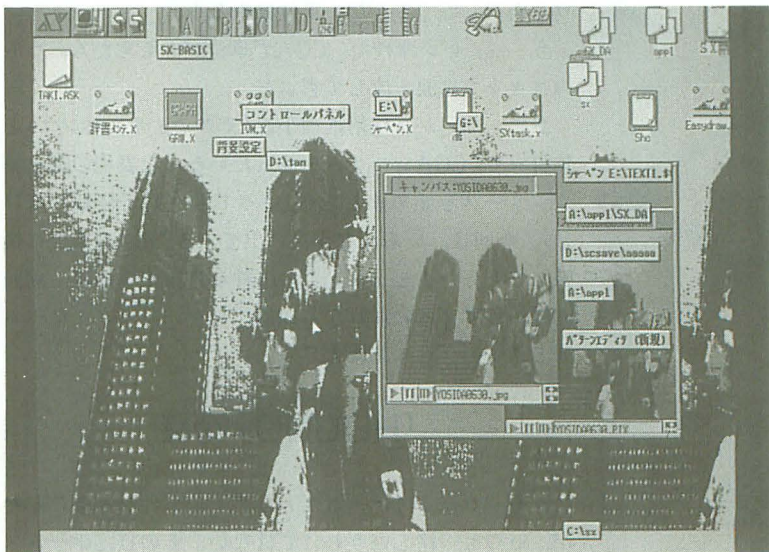



写真2　PICファイルを背景に敷いてみた

さまざまな影

プロジェクトチームDōGA
かまた ゆたか

ホームズの推理の鋭さには，助手のワトスンも感心することしきり。しかし，心の中ではたまには自分も能力を発揮したいと思っているようです。というわけで，今月は2人でいろいろな影の謎に迫ります。

はじめに

先日，大阪デザイナー専門学校で上映会をしました。この学校にはCG科もあるのですが，なんとそこで使われているマシンはX68000(なかにはそれだけの理由で入学される方もいるそうです)。構内を見学させていただきましたが，100台ものX68000が並んでいるのにはかなり圧倒されました。また，すべてのマシンに数値演算プロセッサとハードディスクが装備され，2台に1台はスキャナやカラープリンタまであるというゴージャスさ！

さらにとんでもないのは，このCG科でメインに教えるのは「CGAシステム」なんだそうです。1年間みっちりCGAシステムを勉強した方々が，毎年50人出てくるというわけで……。今年のCGAコンテストが怖いですね。

上映会のあとで懇親会があったのですが，そのCG科の橋本さんがDōGAのスタッフとして志願されました。

かまた「どうです，橋本さん。この際，1，2カ月間ほどDōGAに合宿して，修行をしませんか？」

橋本さん「そんなむちゃな。その間の授業とかはどうするんですか」

CG科の先生「橋本，がんばれ。CGAシステムのノウハウを会得してくるんだ。単位はやる！」

う～ん，いい学校だ。CGAで単位がもらえたら，うちのスタッフは涙を流して喜ぶぞ。

さて，前回から技術推理小説(テクノ・サスペンス)と銘打って，CGAシステムを使ったいろんな表現の謎を暴くシリーズを始めました(といっても次回で終わりだけど)。今回は，いろんな作品に出てくる“影”の表現に挑戦してもらいましょう。この事件を担当するのは，前回同様，世界に名の知れた名探偵，シャーロック・ホームズと助手のワトスン博士です。

「HOUND」の謎

シャーロック・ホームズとは多くの事件をともにしたが，いつも一緒というわけではない。その日，ホームズは別件の捜査に出かけていたらしく，朝から留守にしていた。私は，消えた暖炉の脇の使い古した低い肘掛けイスをテレビの前に持っていき，CGAコンテストのビデオを巻き戻した。

先日は，ホームズに「DRIVIN' WOMAN」の謎を見事に解き明かしてもらったが，たまには些細な謎のひとつやふたつは自分で解決してみたいものだ。さっそく，ビデオの頭からじっくり見てみることにした。

しかし，ついついストーリーなどにのめり込んでしまい，謎らしいものなどさっぱり見つからない。これではいけないと，「DRIVIN' WOMAN」と同じ1カット部門に絞ってみることにした。これならストーリーはないし，高度な技術も盛り込まれている。

特に「OBJECT:MECHANICAL HOUND」(36ページの写真1参照)は，高度な技術がぎっしり詰め込まれている。この作品の謎を解いてやろうと何度も見ていたら，ホームズが帰ってきてしまった。

ホームズ(以下ホ)：やあ，ワトスン君，来ていたのかね。おはよう。おっと，もう昼過ぎか。

ホームズは，事件でも解決してきたところなのか，やけに上機嫌だった。

ワトスン(以下ワ)：ホームズ，私からの依頼で悪いんだが，この「HOUND」の謎を解いてくれよ。

ホ：なんの謎だって？

ホームズは，小雨で少し濡れたコートを掛けながら，私にそう聞き返した。

ワ：「HOUND」の完璧ともいうべきモーションデザインの謎だよ。この作品は，1カット部門ではなく一般部門で応募していたら，技術賞をもらえたんじゃないかな。

ホ：ハッハッハ。残念ながら，その件については特に謎なんかはないよ。CGAマガジン第2号でデータも公開されている。私も分析してみたが，フレームソースの記述などに独自の工夫が凝らされている。よく見てごらん。前足と後ろ足とがまったく同じ形状だろ。実際の犬は骨格が異なり，動き方も違っているんだが，そんなところを省略しているだけだよ。CGAはなにも苦労すればいいってもんじゃないからね。センスのいい手抜きだよ。

ワ：なんだ，そうなのか。それじゃあ，謎ってほどじゃないな。

ホ：ああ，私も最初は，足が地面にめり込んだり，離れたりしないのはどうしてだろうと思ったんだが，「MISSION」のように特別なことをしているわけではない。その証拠に，見る角度を変えて作画したり，地面に模様をつけたりしてやると，微妙にずれたり滑ったりしているのがわかるよ。もっとも，作品としてはそれが目立たないような工夫，たとえば地面に模様を入れないとか，足の先の形状を工夫するとかといった点は実に素晴らしい。作者の下岡さんは，四足動物の歩行について専門的な知識があるだけではなく，CGAに関するセンスがとてもいいんだよ。

今日のホームズはやけによくしゃべる。

ワ：じゃあ，この影はどうだい。ほらっ，雷が光るシーンで一瞬影ができるだろ。

ホームズの動きが一瞬止まった。

ホ：……正直にいうと，その件については僕自身もよくわからない点があるんだ。

ワ：そうだろう，そうだとも。CGAシステムの「REND.X」では，影は発生しないからね。いったい，どうやって影を表現したんだろう。

ホ：ワトスン。君は僕のいったことを少し勘違いしているようだね。あの影は簡単だよ。実に基本的な手法，そう，シャドウポリゴンを自分で生成しているだけだよ。

ワ：シャドウポリゴン？

ホ：いや，そんなにたいそうなものでもない。黒い板を置いているだけなんだ。ほらっ，「SWORD2」を見ればわかりやすい。人体モデルの真下の地面に，丸い黒のポリゴンを置いているだろ(写真２)。これが，シャドウポリゴンさ。半透明のポリゴンを利用することで，模様のある床に影を落とすこともできる。

ワ：でもホームズ，「HOUND」では単なる円ではなく，ちゃんと「HOUND」の形をしていたし，歩くのに合わせて動いていたが。

ホ：同じことだよ。実際にそのシーンを見てごらん(写真３)。ほら，この位置に影ができるということは，真横の斜め上から光が当たっているはずだろう。でも，できている影は斜め上から見たものじゃない。真横から見た形状になっている。これは，ちゃんと計算された影としては不自然だよ。もっとも，注意して見ないと気がつかないがね。

ワ：だから？

ホ：CAD.Xで各パーツを読み込む。そして側面図を見ながら，そのパーツの輪郭をなぞっていくようなポリゴンを１枚作るんだ。細かい凹凸なんかは気にする必要はない。だいたいの特徴がわかればいい。その際，アトリビュートは変えておく。「shadow」とかね。そして，別な名前でセーブして……，ああ，細かく説明していると長くなるが，要するに，各パーツの代わりに黒い１枚のポリゴンを用意して，同じフレームソースの中で，厚みをきわめて薄くしたものを，地面のすぐ上で動かせばいいというわけだよ。

ワ：なるほどね。それじゃあ，君のいう疑問とやらは何なんだい？

ホ：作者は，なぜ雷のシーンだけに影をつけたかだよ。この手法は別にどのシーンでも使えるじゃないか。何か特別な理由があるんだろうか？

ワ：単なる演出の問題じゃないのかい？

ホ：そうかな。あまり長時間影を出していると，不自然なのがわかってしまうからかもしれないし。

ワ：まあ，どっちにしろ，君にとってはたいした謎というわけではないわけだ。

ホ：ん？　何を怒っているんだい。

DōGA法人化への道

“あれっ，DōGAってもう法人化したんとちゃうん？”と思った方もいらっしゃるでしょうが，なんのなんの，それでもこのコラムはまだまだ続きます。これからは，法人化後の動きをDōGAの準スタッフでもある皆さんに報告し，DōGAの今後について，一緒に考えていきたいと思います。

今年の４月に法人化したといっても，法人部門を持つようになったというだけで，プロジェクトチームDōGA自体は，アマチュアの団体として存続しています。アマチュアのDōGAは，例年の活動に加え，今年からCGAマガジンの発行を始めて，あいかわらず活発な活動を続けています。ちょっと，スタッフの高年齢化が進んではいるんですが。

それに対して，(株)ドーガ(こういう表記なのです)は，まだまったく軌道に乗っていません。この会社は，専任スタッフがDōGAの活動を支援するという目的もありますが，社員全員がDōGAの活動(つまり，慈善事業)をしていたのでは，すぐに倒産してしまいます。せめて自分たちの給料分ぐらいは，そして将来的にはDōGAの活動を資金的に援助できるほど，収入を得るぐらいにならないといけません。

収入源としては，従来行っていたような活動はほとんど考えていません。それはあくまでもアマチュアの領域であったからです。ですから，これを読んでいる皆さんは法人部門に直接は関係ないといえるでしょう。

現在予定している収入源は，法人相手の映像制作業務がメインとなっています。つまり，CGAシステムでCGを制作するわけです。はっきりいって，これは得意分野ですね。(株)ドーガの映像制作担当主任は，“EPA2ビデオマニュアル”などで有名な宇宙人森山さんですし，そのほか，わりと名の知れた人たちが，楽しい，あるいはかっこいいCGAを量産することができます。

そこで，従来のCGではコスト的に合わないとか，比較的低解像度でも大量の動画がほしいという市場，たとえばCD-ROMのゲーム，地方テレビ局やCATV，ビジネスプレゼンテーションといったあたりでのCGA制作を実施していこうと考えております。

市場開拓にあたっては，まずサンプルなどを作って，営業活動を行うことが必要でしょう。ところが設立直後に，いきなり，某大企業の知人からビジネスプレゼンテーションの大きな仕事の注文が入り，以来ずっとそれにかかりっきりです。幸先がよいといえばよいのですが，その仕事も今月で終わり，次の予定は入っていません。ということで，来月から，営業活動の仕切直しというところです。

てな感じで，まあ適当にやっていたわけなんですが，ここでちょっと大きな問題が発生しました。それは，次回のお楽しみ。

次回予告：“アマチュアのDōGAが解散に追い込まれる！(ウソ)”

ワ：いや，私としては，せっかく謎を見つけたつもりだったんだがね。こうあっさり片づけられてしまうとね。たまには，私が謎を解くってことがあってもいいと思うんだが。

ホ：それなら，ちょうどよい。実は読者から，ちょっと面白いデータが届いているんだ。この謎に挑戦してくれたまえ。

有川キラー氏の挑戦

淡いグレーの封筒に3.5インチのディスク。ディレクトリをとってみると，「readme.doc」がある。

> シャーロック・ホームズ様
>
> 突然ながら，ひと筆差し上げます。私はCGAマガジン創刊号にデータを載せていただいた，有川キラーと申します。7月号の連載を読み，ぜひともご覧いただきたいCGAがありましたので，お送りいたします。貴殿の素晴らしい推理力をもってすれば，この謎も見事に解決してくださることを疑いません。　草々

ホ：「PIC」というディレクトリがあるだろ。そこに画像データが入っている。「KAGE.TCH」でアニメーションが再生できるよ。

ワ：ファイル名からすると，影に関するCGAだね。

ホ：そのとおり。

短い時間でディスクからメモリに画像データが転送された。どうやら，そんなに複雑な画像ではないようだ。アニメーションはすぐに始まったが，それは，小さな山の上をUFOが通過していくものだった(写真4)。CGAとしてはきわめてシンプルだが，そのUFOは小山に影を落としていた。

ワ：うーん，すごい。問題はこの影をどうやって表現したか，だね。

ホ：そのとおり。

REND.Xにこのような影を発生させる機能がないのは周知の事実だ。いったいどうやったんだろう。見当もつかない。ホームズは横でニヤニヤ笑っている。

ワ：ホームズ，君はもうこの謎を解いたのかい？

ホ：ああ，もちろん。だが正直な話，推理と呼べるようなものではないよ。この手法は以前に私が思いついて，そのためにREND.Xを一部変更してもらったものなんだ。だから，私にとって一目瞭然なのは当然さ。

一目瞭然とまでいわれると，少し腹が立つが，やっぱりわからない。何か手がかりはないものだろうか。コマ送りで1コマずつ見てみる。

ワ：「HOUND」と同じように黒い板を置いたというわけではないね。地面には凹凸があるから。

ホ：そこがいちばんの謎だね。同時にこの手法の最大のメリットでもある。単に地面だけでなく，家や木があっても問題ないんだ。ほかに気がついた点はないかい？

ワ：う～ん，影がぼやけている。

ホ：そのとおり。このような半影も，シャドウポリゴンではない証拠だね。

ワ：たったこれだけの情報でわかるもんか。だいたい，私はいままでCGAに影をつけたことすらないのに。

ホ：おいおい，妙なことをいってもらっては困るね。その理論だと，殺人事件を解決する私は殺人の経験が豊富ってことかい？

ワ：う～ん，それはそうだ。地形を構成する各面の明るさ自体が変わっているなあ。地面のアトリビュートを1フレームごとに変えているとか？

ホ：いっている意味がよくわからないが，全然違う。そんなことするぐらいなら，ペイントツールで影を描くほうがましじゃないかね。

ワ：何かヒントはないかい。

ホ：影で難しいことはなんだい。「HOUND」の影で君自身いっていただろう。形だよ。

ワ：形？　この影の形は単純な丸のようだが。でも，それは飛んでいるこのUFOが丸いからだろう。

リスト1

```
( mov ( 200 ¥div( -500, 500, 1, 20, fno )¥ 300 )  obj   UFO
  light spot ( rgb ( -0.75 -0.75 -0.75 ) -1.0 2.0 -7.0    0.3 )
)
```

各読者連絡事項

>ビデオ配布終了のお知らせ

第5回CGAコンテスト収録ビデオ配布は，おかげさまで今年も大好評のうちに，無事終了いたしました。“ちょっと，マッター！　私のところにはまだ来てないぞ！”なんてことがありましたら，至急ご連絡ください。その際は，現金書留の控えのコピーなどを添えていただければ，すみやかに調査できます。

>シャープ　ショールームCG上映会のお知らせ

東京市ケ谷のシャープショールームに，「映像アトリエ」なるミニシアターが開設されたそうです。そこで，9月9日(木)に上映会を行うこととなりました。

内容は，CGAコンテストのスペシャルセレクションや，DōGA内の秘蔵(？)作品の上映，解説を行います。また，X68000をハイビジョンプロジェクタに直結しての実験上映なども企画しています。

参加ご希望の方は，下記の要領で申し込んでください。120名の皆様を抽選でご招待いたします。

会場：JR・地下鉄市ケ谷駅北西徒歩3分
シャープ市ケ谷ビル1階東京ショールーム“映像アトリエ”

日時：9月9日(木)
1回目 15：00　2回目 18：30

応募方法：官製ハガキに「CG上映会」と明記し，希望上映時間，郵便番号，住所，氏名，年齢，性別，職業，電話番号を記入してください。

応募先：〒160　東京都新宿区若葉1-6
文化放送開発センター内
「シャープ液晶ミニシアター」Oh!X係

ホ：そうとはかぎらないよ。丸い影しかできないから，UFOを丸くしたとも考えられるだろ？
ワ：なるほど。ほかには？
ホ：いままでのCGA作品の中で，同じような雰囲気のカットがあるものは思い当たらないかい？
ワ：そうだね，こんな影が落ちている作品なんかあったかな。
ホ：いや，影としてではなく。別な色で。
ワ：別な色？　赤とか，青とか……。そういえば，第2回のCGAコンテストのオープニングで，エンタープライズ号が光子魚雷を撃つシーンがある！　光子魚雷が船体をかすめるとき，赤い光が映りこんでいた。
ホ：そう，そのとおり。あれはどうやっていた？
ワ：点光源だよ。えっ？　ということは，もしかしてこれは黒い点光源……。まさか！
ホ：だいたいそれで正解だよ。正確にいえば，点光源ではなく，スポット光源だけどね。
ワ：ちょっと待ってくれ。黒い光で照らすとは，どういうことなんだい。いまいちピンとこないが。
ホ：フレームソースでこう記述するんだよ(リスト1)。UFOと同じ位置にスポット光源を斜め下向きに置き，UFOと一緒に動かす。ポイントは色の指定をマイナスの数値にすることさ。
ワ：CGA共通規格では光の色の指定は0〜1となっていたはずだが。REND.Xでよくエラーにならないね。
ホ：昔のバージョンではエラーになっていたんだけど，この影の表現のために小林氏に直してもらったんだ。
ワ：なるほど，推理する前から“REND.Xにはこのような機能はない”と決めつけたのがいけなかったんだね。しかし，君もいっていたように，この手法では丸い影しかできないのが欠点だね。
ホ：ああ，細長い形なら複数のスポット光源を置くといったことも可能だけど，複雑な形や正方形なんかは表現できないね。

陽の当たる場所

ワ：いや，しかしなんだね。自分で推理するというのもなかなか面白かったよ。
ホ：そうかい，じゃあ私個人からもちょっとした問題を出させてもらっていいかな。
ワ：ああ，どうぞ。喜んでお受けいたしましょう。
ホ：では，このアニメーションを見てくれたまえ。
　それはCGAマガジン創刊号からF1のデータを流用したものだった。ピットからF1が出てくる(写真5)。
ワ：なかなかかっこいいカットじゃないか。しかし，問題の影はF1にはついていないんじゃないかい？
ホ：いやいや，今回の謎はさっきのような影とは違うよ。F1が暗いガレージから，日光が当たるところに出てくるだろう。そこが問題なんだ。
ワ：えっ，ガレージの中では光源を暗くして，表に出ていくにしたがって明るくするのではだめなのかい？
ホ：おいおい，それだけなら，車体全体が同時に明るくなっていくだろう。よく見てくれよ。ひとつの車体の中で影の部分だけが暗く，光が当たっている部分だけが明るいだろう。
ワ：本当だ。きっちり分かれている。まさか車体を前後に分けて作って，アトリビュートを変えていっているのか。でも，車体が動くと境界線も変わっていくしなあ。
ホ：この境界線，いや境界面は適当にごまかしたものじゃないよ。正確に斜めに切れている。
ワ：……境界面。わかった！　簡単じゃないか。ほら，ここ。この境界面のところに斜めの半透明の黒い板を置いているだけじゃないか。
ホ：いやいや，そんな単純な……。待てよ。なるほど，確かにそれでもできそうな気もするな。
ワ：おいおいホームズ，君らしくもない。問題を出したほうがそんなんじゃあ困るよ。
ホ：面目ない。君のアルゴリズムには何か問題があるような気がするが，論より証拠だ。やってみよう。
　ホームズは，FFE.Xを起動すると，メモを見ながら車体を置いたり，視点を設定したりした。そして最後に，1枚の大きな板を斜めに置いた。
ホ：こんな感じかな。
ワ：レンダリングしてみよう。
　私はわくわくしながら，計算が終わるのを待った。
ワ：どうだい，ホームズ！　だいたい，それらしくできたじゃないか(写真6)。
ホ：う〜ん，確かに同じような絵になるね。しかし，この方法の問題点もはっきりした。
ワ：なんだい，負け惜しみじゃないだろうね。
ホ：いや，とんでもない。簡単なことさ。視点が影の部分にあるようなときに困るんだ。ガレージの中から見ているようなカットを作ってくれたまえ。
ワ：え？　視点の位置を変えるだけだろ。簡単じゃないか。まったく同じようにできるに決まってるよ。
　私は，ホームズと席を替わり，さっそく作業に入った。FFE.Xを起動し，視点の位置を変更して，セーブした。
ワ：あっ，そうか。今回は手前が暗くて奥が明るいから，黒の半透明ではなくて，白の半透明にしなくちゃいけないんだな。でも，それもアトリビュートファイルを修正するだけだ。
　ホームズは何もいわなかった。
ワ：さあ，これでいい。
　しかし，レンダリングの末に出てきたものは，陽の当たる場所に出ていくというよりは，煙か霧の中に突っ込んでいくような画像だった(写真7)。
ワ：あれっ，こんなはずじゃ。透明度の割合がまずかっ

たのかなあ。
ホ：いや，何度やっても無駄だよ。確かに暗いことと黒いことは似ているが，明るいことと白いことは異なるという点に気づいていなかったんだよ。
ワ：じゃあ，君はどうやったんだい？ だいたい，君の方法では視点が影の中の場合でも，ちゃんと表現できるというのかね。
ホ：ああ，もちろん。
そういってホームズが見せてくれたアニメーションは，完璧なものだった(写真８)。
ワ：降参だよ。どうやったか教えてくれたまえ。
ホ：境界面に斜めの大きな板を置くというのは，ほぼ正解だよ。ただ，私は影の部分と陽の当たる部分でレンダリングを２回に分けて，それをPILE.Xで合成してあるんだ。
ワ：もう少し詳しく説明してくれないか？
ホ：これを見ればすぐわかるよ。まず，１枚目は光源が明るい状態でごく普通にレンダリングする(写真９)。そして，もう１枚は光源を暗くした状態の画像をレンダリングする(写真10)。
ワ：なんだい？ この真っ赤な部分は？
ホ：境界面に置いた板だよ。
ワ：また，ずいぶんと派手な色だね。周りと調和が取れてないよ。
ホ：そうじゃないと困るんだよ。この画像をREP.Xに呼び込む。
ワ：REP.Xというのは？
ホ：画面上の色を置換するツールさ。このツールで真っ赤な部分を黒(透明色)に置き換える。こうすると，影の部分だけの画像ができるだろう(写真11)。これを陽の当たる画像のほうに重ねてやれば出来上がりだ。
ワ：なるほど。
ホ：注意するポイントとしては，さっきもいったように境界面の板はほかの部分で使われていない色にし，アトリビュートのアンビエントのパラメータを１，そのほかのパラメータを０にしておくこと。
ワ：確かに。そうでないと，REP.Xでちゃんと置換できないからね。
ホ：そのとおり。それから，レンダリングの解像度は512にして，アンチエイリアシングは使用しないこと。
ワ：えっ，なぜだい。私は普通256×256のアンチ２倍を使用しているんだが。
ホ：アンチを使うと境界面の周辺がぼやけて，境界面とは微妙に違う色ができるだろ。そうすると，REP.Xで置換されないんだ。
ワ：でも，512の画像はアニメーションするには負担が大きすぎるけど困らないのかい。
ホ：いやいや大丈夫。2枚の画像を合成したあと，DCHANGE.Xでアンチをかけながら256に変換すればいい。
ワ：でも，この光と影の手法は面白いね。いろいろ使えそうだよ。格納庫の上のプールが開いていって，中のロケットに光が射し込んでいくシーンなんか，かっこいいじゃないか。
ホ：……。

夫婦でQ&A

うさ子：毎日のようにお手紙をいただき，ありがとうございます。
ゆたか：しかしながら，当方の管理上の都合(ビデオの申し込みと手紙をいっしょに入れた場合など)で，封筒とお手紙がばらばらになることがよくあります。お手紙のほうにお名前，住所がないときには，こちらから連絡がとれなくなってしまいます。
うさ子：ご面倒をかけて，もうしわけありませんが，お手紙のほうにも連絡先を書いていただきますよう，お願いします。
＜Yさん＞CGAマガジンで楽しんでいます。しかし，みんなはあの「膨大」な画像データを，どーやって管理しているんでしょう。私がCGA用に用意した80Mバイトあるハードディスクも，すでに40Mバイト埋まってしまいました。データがほかのドライブにもあふれだしたので，LHAなどで使わないデータを圧縮したんですけど，それでもダメです。みんなはどーしてるんだ！ 教えてくれ～！
うさ子：確かに，あの大量の画像データを残しておくのはたいへんなことですね。
ゆたか：DōGAでは，昔はフロッピーディスクに残していたんですが，とてもじゃないので，最近はもっぱらMO(光磁気ディスク)です。ハードディスクと比べて，アクセススピードが遅いのが難点ですが，１枚で128Mバイトという大容量が魅力です。最初にドライブさえ買ってしまえば，あとは5,000円でハードディスクを１台増設できるようなもんですから。
うさ子：でも，こういうお手紙もありますよ。
＜Nさん＞CGA用にMOを購入しました。しかし，買ったばかりのMOが，みるみるいっぱいになっていきました。すげえ，すげえ。
ゆたか：やっぱり，ビデオに録画して，データのほうは消すしかないのかな～。
うさ子：全部残すのではなく，好きなとこだけ寄せ集めるとか。
ゆたか：そうだね。TCHED.Xで編集すると，よい勉強になるかもしれないし。
＜Sさん＞DōGAの一部法人化，おめでとうございます。世間の荒波にもまれることもあるでしょうが，「４×８＝48」や「お役人様」にも負けずガンバッテください(細川ふみえ風に)。とりあえず，私のできることとして，カンパをさせていただきました。“CGAの普及”にお使いください(別に地下の秘密基地建設に使っても，謎の最新兵器で世界征服を行う軍資金に使っても結構です)。
うさ子：ご声援ありがとうございます。皆様からのカンパは有効に活用させていただきます。
ゆたか：ただし，Sさんのカンパはご本人の希望により，世界征服のための軍資金とさせていただきました。残念ながら，あと22兆9999億9999万９千円ほど足りませんので，とりあえず定期預金しておきます。ところで，細川ふみえさんって，だれ？
うさ子：あっ，私は知ってる。胸の大きな女の人でしょ……。
ゆたか：……。
＜？さん＞私は臨床工学技師で，人工透析などに携わっております。いままではX1turboを使っていましたが，これからはX68000で心臓のシミュレートをしたいと考えています。データも放射線科から，RIで得られたものをわけてもらおうと思っています。最適なシステムを教えてください。
ゆたか：最適もなにも，X68000でCGアニメーションするならの話ですよね。
うさ子：じゃあ，ほかのシステムなら？
ゆたか：Indigo²にPRISMSなんかを載せたら，きっと素晴らしいものができますよ。
うさ子：なんですか，それは？
ゆたか：詳しくは知らんけど，こないだの“CG OSAKA”で見かけてん。1000万円はくだらんやろ

完璧な影を求めて

ワ：しかし，どの手法にしても，正確な影を落とすことはできないんだね。

ホ：確かに，これらの手法ではね。

ワ：なんだい？　まるで，完璧な影が作れるみたいな言い方だね。

ホ：ああ，そういっても過言じゃない。

ワ：まさか？　レンダリングアルゴリズムを変えるとかいうんじゃないだろうね。

ホ：もちろん，現在あるツールだけで可能だよ。試しに，何か３次元の物体を作ってくれないかい。

ワ：ああ，いいとも。

ホ：……そうだ。ついでに，その物体のどこかに半透明の面を使ってくれたまえ。

ワ：半透明？　君のいう手法は半透明の面がないとできないのかい？

ホ：いや，関係ないよ。ただ，半透明の面があると，影がステンドグラスのようになって，きれいじゃないか。

最初私は，ホームズが冗談をいっているのかと思った。そこまで完璧な影ができるとは思わなかったからだ。

ワ：本当にいいのかい？

ホ：ああ，もちろん。どうぞ。

私はさっそくCAD.Xに向かった。ステンドグラスのようなということで，簡単な色眼鏡を作ってみた。

ワ：少しおかしいけど，こんなもんでどうだい(写真12)。

ホ：十分だよ。作業にはちょっと時間がかかるんだが。

そういって，ホームズは作業に没頭した。しかし，なかなかうまくいかないようだ。ぶつぶついいながら，何度も作画をやり直し，作業は１時間近く続いた。

ホ：まだちょっとおかしいような気もするが，だいたいこんなものかな？

ワ：ほう，どれどれ。

覗き込むと，画面にはタイル状の床と眼鏡，そしてその眼鏡の完璧な影がタイルに落ちている様子が表示されていた(写真13)。

ワ：あっ，すごいじゃないか。どうやったんだい？

ホームズは答えない。

ワ：マイナスの光源を使ったのか……。でも，それだとこんなに眼鏡の形がはっきり出るわけはないし。

ホームズがにやにやする。どうやら的外れなことをいってしまったらしい。

ワ：ちょっと映り込みに似ているなあ。床が半透明で，その下に同じものがうまく置いてあるとか？

ホ：違っているが，少しは関係あるかな。

ワ：これは画面の右斜め上から光が射しているんだね。でも，斜めから当たっている分だけ，ちゃんと影が歪んでいるなあ。どうやって，歪ませるんだ。なにか，新しいツールかい？

ホ：いや，何も新しいツールなど使ってはいないよ。でも，この画像では歪みがわかりにくいかもしれない。このほうがいいかな。

ホームズがポンとリターンキーを押すと，眼鏡がクル

う。

うさ子：はいはい。

ゆたか：でも，CGAコンテストにリアルな心臓のアニメーションが出品されたら面白いね。

うさ子：うっ，気持ち悪いのきらい～。

ゆたか：ぜひ技術的なサポートを行いたいと思うのですが，連絡先がわかりません。すみませんがもう一度お便りください。ところで，RIってなんだろう？

うさ子：あっ，私，知ってる～。

ゆたか：ちょっと待て。RADIOACTIVE(放射能) INTERACTION(相互作用)。

うさ子：ちょっと違います。

ゆたか：“I”はローマ数字かも。Ｉ次冷却水。こ，怖い……。

うさ子：全然違います。RADIOACTIVE(放射能) ISOTOPE(同位体)。あれ？

ゆたか：意味が通らんから，違うと思うで。

<Ｔさん>CGAマガジンの説明で少しわかりにくいところがありました。展開のところで，“２Ｍ用”とありましたが，これは２Ｍの人ですか？　それとも，２Ｍ以上のマシンという意味なのでしょうか。

ゆたか：Ｔさんのお手紙で少しわかりにくいところがありました。“２Ｍの人”とは，身長が２メートルある人ということですか？　紳士服メーカーにお勧めということですか(そりゃあ，３Ｍだって)。

うさ子：こらこら，あげ足をとってどうするんですか。この場合，２Ｍバイト以上のメモリをお持ちの方という意味です。ほかのソフトでも，メモリが多すぎるためにソフトが動かないということは，まずありません。

ゆたか：メモリをお持ちでも，ちゃんとマシンにセットしないといけません。

うさ子：あげ足はもうよろしい。

<Ｓさん>「CGAマガジン」は，「CGAシステム」を使えない人間にもちゃんと使えるのでしょうか？

うさ子：絶対に大丈夫です。

ゆたか：対象年齢は猫から成人までです。

うさ子：にゃ～にゃ～。

<GT-Rさん>奈良県に「TAKERU」はあるのでしょうか。460円も出して日本橋まで行ったのに，CGAマガジンが発売延期になっていました。腹立つ～。ところで，CGAマガジン第２号のマウスポインタが増殖するバグはなんとかなりませんか。私はバックアップを取って，Ko-SHELLをD-SHELLに代えて使っています。グラフィックが表示されませんが，スクロールなどのレスポンスが速くていいです。

ゆたか：すいません。ごもっともです。発売日については多少変動もありえますので，定期購読をお勧めします。

うさ子：奈良県では，上新電器の学園前店，奈良一番館，大和郡山インター店に設置されています。全国約300店に設置されていますから，各都道府県にあるはずです。詳しくは，ブラザー工業TAKERU事務局　☎052(824)2493にお問い合わせください。

ゆたか：マウスポインタのバグは，68030に対応したために発生したものです。原因はすでにわかっているのですが，著作権に抵触する問題なので，現在対応中です。第３号に間に合うかどうかは微妙です。

うさ子：D-SHELLに代えるのはよい方法かもしれませんね。でも，第３号からはKo-SHELLも，皆さんのご意見をもとにバージョンアップします。お楽しみに。

<？さん>(５月号のこの連載を読んで)かまたさん，結婚おめでとうございます。でも，相手は誰だ。(約３分後)ナニィ，許せん！　この野郎，いちゃついてんじゃねえ。チクショー！

うさ子：その節はたいへん失礼いたしました。

ゆたか：反省，反省。やっぱり，公共性のある誌面上ではわきまえないといけませんね。なっ，うさ子。

うさ子：そうですね，ゆたかさん♥

ゆたか：その“♥”は不要だよ♥

うさ子：いや～ん，バカン♥

ゆたか：なにい。コ，イ，ツー♥

(ほかのスタッフから読者の皆さんへのご注意：以後，この２人にいちゃつく口実を与えるようなお手紙は，固くお断り申し上げます)

クルと回りだした。そして，それにちゃんと影もついて回っている。

ワ：おお。ちょっと待ってくれ。これはアニメーションにも対応しているのかい？

ホ：当然じゃないか。静止画でしか使えない手法など，CGAの役に立たないよ。

思わず私はスペースキーでアニメーションを一時停止し，さらにコマ送りした。

ワ：間違いない。ホームズ，この影は刻々と形を変えているじゃないか！

ホ：そりゃあ，そうだよ。光源が一定で物体が回転すれば，その影の形は変わるよ。

ワ：でも，どうやったらそんなことができるんだい？だって，1フレームごとに形が変わるんだぞ。作りようがないじゃないか。まさか……。

ホ：まさか，1フレームごとにモデリングしたとでもいいたいのかい。とんでもない。そこまで暇じゃないよ。つまり，この影はシャドウポリゴンではないってことさ。

ワ：もったいぶらずに教えてくれよ。私がいくら考えても，わかりっこないよ。

ホ：ハハハ，ずいぶん簡単にあきらめたね。でも，原理はわりと簡単なんだ。前回も似たようなテクニックを使ったよ。

ワ：前回とは「DRIVIN' WOMAN」のことかね。何を使ったっけ？　映り込みの正体はマッピングだったという話だっだよね。もしかすると，これもマッピングなのか。

ホ：そうだよ。それで正解さ。1フレームずつ影の形状が変わるのは，映り込みが1フレームずつ変化したのと同じだよ。まず，最初の影の画像を動画で作って，その画像を1枚のポリゴンにマッピングして，眼鏡の下に置いているだけさ。

ワ：なるほど，いたって単純だな。でも，どうやったら，正確な影の画像なんてのができるんだい？

ホ：それが最大の謎さ。この図を見てくれ(図1)。ここに光源があって，床に影が落ちている。もし，光源の位置に視点があって，物体の方向を見ていたとしたら，どうなる？

ワ：どうなるって，どういうことだい？

ホ：ほら，影は物体に隠れてまったく見えないだろ。

ワ：ああ，そうだろうね。光源の位置から見えるところには光が当たるんだから。

ホ：そうさ。つまり，影の形状とはその物体を光源の位置から見た形と同じになるんだ。

ワ：えっ，そうなのかい。なんか，ちょっと違うような気もするんだが。

ホ：確かに違う。特に光源が低い位置にあるときはね。なぜなら，光源から見た図は，物体の形を視線に垂直な面に投影するのに対し，影は水平な面に投影したものだからね。でも，それは簡単に修正できるんだ。これが影，こちらが光源から見た図とすると(図2)，ほら，画面の縦方向に引き伸ばすとまったく同じになる。どれだけの割合で引き伸ばすかは，光線ベクトルのX，Y，Z成分から求められる。えっと，式はこうだな。

$$\sqrt{X \times X + Y \times Y + Z \times Z} : Z$$

どうして，画面の縦方向に引き伸ばすとまったく同じになるのかは疑問だった。ただ，それ以上聞いたってわかりっこなさそうだ。

ワ：ああ，わかった。光源から見た動画を作って，その動画をマッピングした板を置いてやると影になるというわけか。結構，簡単だね。

図1　光源と影

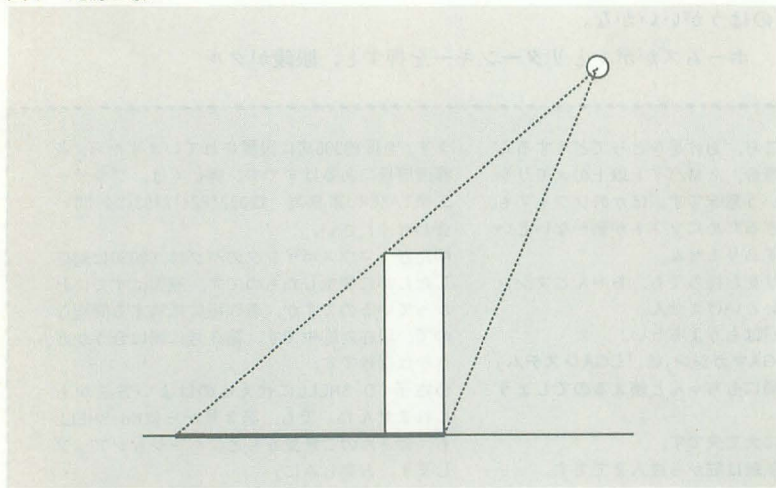
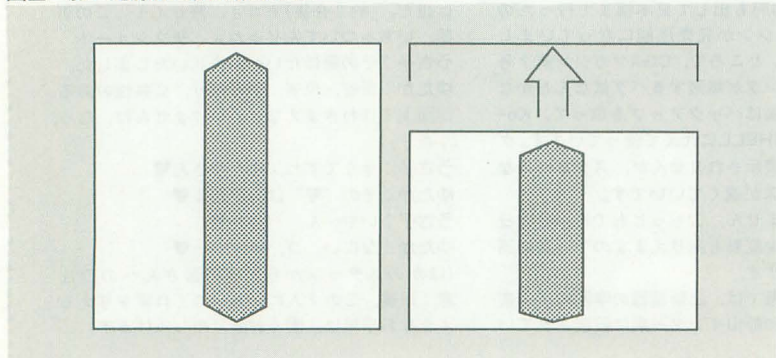



図2　影と光源から見た形の違い

実際の作業

ホ：確かに，理論的には簡単なんだが，やろうとするとたいへんなんだ。実際にやってみよう。

ワ：お手やわらかに頼むよ。

ホ：まず，用意する形状データは，眼鏡，床，影のポリゴンだ。

ワ：影のポリゴンというのは，影の画像を張り付ける1枚の板だね。

ホ：そう。眼鏡よりひと回り大きな正方形でいい。この

眼鏡，「glass.suf」はX，Yの値が±300ぐらいだから，±500もあればいいだろう。それではまず，影となる画像を作ろう。フレームソースはこんな感じになるだろう（リスト2）。

ワ：ずいぶん簡単なフレームソースだね。特に注意する点はないのかい？

ホ：いくつかある。まず，視点の位置だが，平行光源の位置は無限遠だから，光線ベクトルのX，Y，Zと同じ比率でできるだけ遠い座標に設定する必要がある。

ワ：ちょっとわかりにくいが。

ホ：つまり，この例では光線ベクトルは（－3，－2，－1）だろう。だから，視点のX座標，Y座標，Z座標の比は，3：2：1ということさ。できるだけ大きくといっても30000ぐらいが適当かな。

ワ：それで，この例では，（30000，20000，10000）なんだね。

ホ：次に画角だが，これはかなり小さくしておかないと，できる画像が小さくなりすぎてしまう。

ワ：この例では1.5度となっているが，数値はどうやって求めたんだい？

ホ：単なる試行錯誤さ。作画させてみて，目的の物体がだいたい画面いっぱいになるぐらい。そして，どのフレームを描画しても，決して画面から出ないように注意する必要がある。一部でもだ。

ワ：そいつは，ちょっと制限が多いな。この例のように1カ所でクルクル回るのならいざ知らず，もっと大胆に動けば，画面から簡単にはみ出してしまうよ。

ホ：いや，位置が動く場合はその物体をtargetで追いかければいい。もちろん，その場合は視線を常に平行にしておく必要があるから，視点の位置も物体と同じように動かないといけない。うん，そうだね，こんなふうに記述したほうがいいかもしれない（リスト3）。

ワ：なるほど，参考にさせてもらうよ。それで，フレームソースができたら，レンダリングすればいいんだね。

ホ：いや，その前にアトリビュートファイルを用意しなくては。

ワ：そりゃあ，そうだが，普通のアトリビュートでいいんだろ？

ホ：いや，なんといっても影だからね。半透明の部分以外は黒にしておく必要がある（リスト4）。半透明のところの色はそのままで，属性はもちろん半透明にして，環境光を大きく設定しておく。

ワ：これでステンドグラスのような色のついた影ができるわけか。

ホ：用意ができたら，作画させてみよう。

いつものように作画が始まった。ワイヤーフレームで眼鏡の外形を描写したあと，画面上部から色がついていく……，はずが真っ黒になっていく。

リスト2 KAGE.FSC

```
#frame( fno, 1, 20 )
@5.3@
fram(    light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -1.00 )
         ( mov (    30000   20000   10000   ) eye deg(  1.5 ) )
         ( mov (        0       0       0   ) target )
         ( mov (        0       0       0   )
           rotz ( ¥fno*18¥ )
           obj   glass
         )
)
#endframe
```

リスト3 KAGE2.FSC

```
#frame( fno, 1, 20 )
@5.3@
fram(    light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -1.00 )
         ( mov (  0  ¥ div ( -500, 500, 1, 20, fno ) ¥  0  )
           obj   glass
                 target
                       ( mov (    30000   20000   10000   ) eye deg(  1.5 ) )
         )
)
#endframe
```

CGAマガジン編集部より

DōGAでは再三，「第3号には素晴らしい特集を！」とCGAマガジン編集部に進言してきた。しかし編集部は，“やる，必ずやる。やらなければいけない”といいながら，時間を理由に「特集なし」との方針を発表した。DōGAではこの暴挙に対して，“不信任案”を提出する見込みである。不信任案が可決されたら，煽りを受けてDōGAが解散になったりして。

<CGAマガジン第3号発表！（のはず）>

さて，CGAマガジン第3号は，「アマチュアCGAコンテスト特集・第2弾」として，「見せますフレームソースファイルの奥義“MACHINE VISUALIZATION”」と「点光源の使い方“FREE WAY”」を。投稿では「あの創刊号のエンジンがパワーアップして再び登場，今度はターボだ」「ここまでくると職人芸（？）チェーン＆スプロケット」「華麗な動きの人体モデル妖精」など。また，第2号に引き続き，音楽データ，バージョンアップツールなどが盛りだくさん。乞うご期待。

<締め切るぞ！　定期購読申し込み>

当初，「定期購読は随時申し込み可能」を考えていたのですが，手間が大きすぎるとの意見が大勢を占め，定期購読の申し込みは今回（第3号から）限りとすることになりました（次回の申し込みは第6号？）。

申し込んではいないけど，ぜひ申し込みたいという方もまだ間に合いますので，下記の方法に従い，お申し込みください。

記

郵便振込，もしくは現金書留で，CGAマガジン定期購読代金として5,000円（1年4回分）をDōGAまでお送りください。入金が確認され，定期購読登録がなされた時点で，定期購読登録確認のハガキをお送りします。1カ月以上たってもハガキが届かない場合には，当方までご連絡ください。

諸注意

・郵便番号・住所・氏名・電話番号を忘れずに記入してください。

・定期購読は第3号からとなります。よって，今回から定期購読された方は，CGAマガジン第3号を購入しないように注意してください。

・製品の性格上，返品には応じられませんが，お申し出があれば定期購読を解約し，残金をお返しいたします。

・3.5インチディスク版をご希望の方はその旨をお書き添えください。

郵便振込みの場合

口座番号　大阪3-109598

加入者名　DOGA

注意

・通信欄に必ず，「CGAマガジン定期購読希望」と記入してください。

・住所欄には必ず，ご自分の住所氏名を記入してください。

郵送（現金書留）の場合

〒533　大阪市東淀川区淡路5-17-2 102号

プロジェクトチーム DōGA内

「CGAマガジン定期購読係」

ワ：おっ，なんかおかしいぞ。画面が真っ黒，いや，レンズのところだけ色がついているが……。

ホ：あわてることはない。影となる画像なんだから，真っ黒でいいんだ。

ワ：でも，この真っ黒の画像を正方形のポリゴンに張り付けても，真っ黒の四角にならないのかい。

ホ：大丈夫だよ。確かに人間の目には真っ黒に見えるが，この画像にはちゃんと眼鏡の形が描かれているんだ。画像ファイルには，黒でも透明ビットが立っている黒とそうでない黒がある。眼鏡がないところは，透明ビットが立っていて，マッピングしたときは何もない状態になるんだ。

ワ：何もない状態？

ホ：この一見真っ黒の画像が「kage001.pic」だ。それと真っ白の画像「siro.pic」を用意する。そして，この2枚の画像を重ねてみる。

slide /o siro.pic kage001.pic

とすると，ほら，眼鏡の形が現れた(写真14)。

ワ：なるほど，何もないところは下の色が出るんだ。

ホ：そういうこと。納得してくれたら，レンダリングを終わらせてしまおう。

しばらくの間，画面にはほとんど真っ黒の画像が作画されつづけた。

ワ：これで20枚の影となる画像ができた。

ホ：次に，床だけの画像を1枚作画させよう。

ワ：なんだい，それは？

ホ：影のポリゴンは物体の下，Z＝0の位置に置くだろう。そうすると，床と同一平面になってしまう。それではレンダリングできないから，床だけ先にレンダリングした画像を作っておいて，眼鏡と影を作画させるとき，背景を読み込ませて合成するんだ。

ワ：わかった。床の大きさ，模様は適当でいいだろ。こんなもんかな(リスト5，写真15)。

ホ：それでいい。そのほかにしておくのは，影のポリゴンをマッピングできるようにuv座標をつけ加えることだな。

ワ：AMAP.Xを使うわけだ。

ホ：それでもいいが，今回の場合は1面だけだから，エディタで書いたほうが早いだろう。AMAP.Xでは，同じ正方形でもCAD.Xで作成した頂点の順番によって，画像の張り付き方が変わってしまうんだ。「KAGE.SUF」と「KAGE.ATR」はこうなる(リスト6，7)。

ワ：これで準備はすんだ。

ホ：最後に，メインとなるフレームソースを書くだけだな(リスト8)。

ワ：影のポリゴンを置くところで，何か複雑なことをしているね。

ホ：ああ。それ以外は何も問題ないよ。

ワ：じゃあ，順番に聞くよ。vec(320001)とは，いったい何のことだい？

ホ：vecはベクトルで方向を与える命令だ。つまり影のポリゴンを，影が伸びる方向に向かせているんだよ。影の伸びる方向というのは，光源の方向，つまり光源ベクトルのX，Yの値でわかる。一般に，

リスト4 glakage.atr

```
atr  glass1 {   col ( rgb ( 0.00 0.00 0.00 ) )
                tra ( 0.00 )
                amb ( 0.00 )
                dif ( 0.00 )
                spc ( 0.00 0.00 0.00 )
        }
atr  glass2 {   col ( rgb ( 0.00 0.00 0.00 ) )
                tra ( 0.00 )
                amb ( 0.00 )
                dif ( 0.00 )
                spc ( 0.00 0.00 0.00 )
        }
atr  lensL {    col ( rgb ( 0.05 0.35 1.00 ) )
                tra ( 0.50 )
                amb ( 1.00 )
                dif ( 0.00 )
                spc ( 0.00 0.00 0.00 )
        }
atr  lensR {    col ( rgb ( 0.90 0.15 0.35 ) )
                tra ( 0.50 )
                amb ( 1.00 )
                dif ( 0.00 )
                spc ( 0.00 0.00 0.00 )
        }
```

リスト5 back.fsc

```
#frame( fno, 1, 1 )
@5.3@
fram{   light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -1.00 )
        { mov (    2000   -750    1000  ) eye deg( 60 )
        }
        { mov (       0      0       0  ) target
        }
        { mov (       0      0       0  ) obj yuka
        }
}
#endframe
```

リスト6 KAGE.SUF

```
obj suf kage {
atr kage
prim uvpoly (   -500    -500    0       0       0
                        -500    500     0       255     0
                         500    500     0   255         255
                         500   -500     0       0               255
        )
}
```

リスト7 KAGE.ATR

```
atr  kage {     col ( rgb ( 1.00 1.00 1.00 ) )
                tra ( 0.50 )
                amb ( 1.00 )
                mapwind (  0  0 255 255 )
                mapview (  0  0 255 255 )
                mapsize (  0  0 255 255 )
                colormap ( kagemap.pic )
        }
```

リスト8 main.fsc

```
#frame( fno, 1, 20 )
@5.3@
fram(   light pal( rgb ( 1.00 1.00 1.00 ) -3.00 -2.00 -1.00 )
        { mov (    2000   -750    1000  ) eye deg( 60 ) }
        { mov (       0      0       0  )  target       }
        { mov (       0      0       0  )
          rotz ( ¥fno*18¥ )
          obj   glass
        }
        {
          vec ( 3 2 0 0 0 1 )
          scal(     0.9    0.9  0.9   )
          scal(     3.74   1     1 )
          scal(     0.75   1.0   1.0   ) /* 画面比補正 */
          obj kage
        }
}
#endframe
```

リスト9 KAGE.TXT

```
#frame ( fno, 1, 20)
copy kage$fno$.pic kagemap.pic
rend /a2 /g /hback.pic /t glass.* kage.* main.fsc -s$fno$:$fno$
```

vec（−X−Y０００１）
となるんだ。
ワ：なぜそうなるかは聞かないけど，もしかしてこのあたりを工夫することで，光源が移動していくアニメーションなんかもできるんじゃないかい。
ホ：そのとおりだよ。面白い使い方ができるかもしれないね。
ワ：以下，３行連続してscalが続くけど，これは１行にまとめられるんじゃないかい。
ホ：それはそうだが，それぞれ意味が違うから，別々に記述したほうがわかりやすいのさ。
ワ：では，scal(0.9 0.9 0.9)の意味は？
ホ：大きさ合わせとでもいうかな。影のポリゴンの大きさは，今回は１辺を1000にしたけど，別に800でもいい。でも，そうすると影の大きさは変わってくる。本当の影の大きさになるように調節するためのスケールさ。
ワ：この値はどうやったら求められるんだい？
ホ：影の画像を作画させるときの画角なども影響するし，残念ながら簡単には求められない。試行錯誤で求めるしかないね。
ワ：それは面倒な作業だね。それじゃあ次の，scal(3.74 1 1)は？
ホ：これはもう解説しただろ。実際の影と光源から見た図は投影面の違いから縦方向(X方向)の長さが異なってくるんだ。これを修正するための値で，

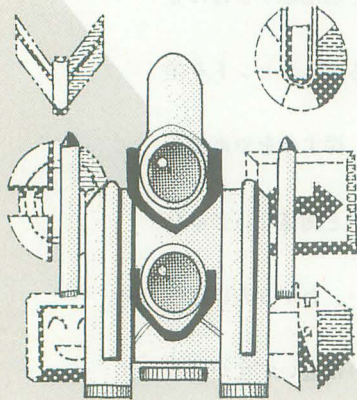
$$\frac{\sqrt{X\times X+Y\times Y+Z\times Z}}{Z}$$

となる。
ワ：Xが３，Yが２，Zが１だから，３×３＋２×２＋１×１＝14。$\sqrt{14}$＝3.74，3.74÷１＝3.74ということか。それじゃあ，最後のscal(0.75 1 1)は？
ホ：これは簡単。影となる画像を先に用意するが，その画面の縦横比は３：４だよね。これを正方形に張り付けると，少しだけ縦に引き伸ばされることになる。それを補正するための値さ。
ワ：３：４だから，0.75なんだね。それと，もうひとつ疑問があるんだが，物体が動いていく場合はこの影のポリゴンもそれに合わせて動かす必要があるのだろう？
ホ：もちろんそうだよ。
ワ：だったら，眼鏡が回転する場合でも，それに合わせて影のポリゴンも回転させる必要はないのかい。
ホ：いや，それはちょっと違う。だって，影の画像を作る過程で眼鏡はすでに回転しているじゃないか。回転している影の画像を作ったのだから，影のポリゴン自体は回転しなくていいんだよ。
ワ：なるほど。これですべて揃った。レンダリングだ。
ホ：いや，その前につまらない作業をひとつ，
REN BACK001.PIC BACK.PIC
を実行しておこう。
ワ：おお，よくやるミスだね。こうしておかないと，動画の背景を読み込むのと勘違いして，２フレーム目以降でエラーになってしまう。
ホ：そういうこと。解説は終わりだ。RENCON.Xを使って作画してみようじゃないか。RENCON.Xの制御ファイルはこうなる(リスト９)。
作画実行は，
RENCON KAGE.TXT
これで完成だ。
作画を始めると，きちんと画像が生成された。
ワ：しかし，この手法は結構たいへんだな。
ホ：……確かに，そのとおり。
ホームズは立ち上がり，窓辺に立った。
ホ：しょせん，こんなものはただのまやかしにすぎない。単なる私の自己満足さ。
ワ：そこまでいうことはないよ。確かに，長編作品ですべての物体に影をつけるのはたいへんだろうけど，４カット部門の作品なら十分可能さ。
ホ：でも，影のために作品を作るなんてことはナンセンスだよ。
ワ：まあ，そうだけど，通常の作品制作中に，このカットでは影をつけたいとか，このカットでは影が重要な意味をもつというときに，それを表現する方法があるというのは，決して無意味じゃないよ。
ホ：そうだね。そういった作品が発表されることを願うよ。本当に……。
いつのまにか雨は止み，薄日が射していた。ふと，床に目をやると，そこにはロンドンの街をじっと見おろす男の影がたたずんでいた。

終わりに

さて，７月号のこのコーナーで出題した懸賞つきクイズで，「正解は10月号に」とありましたが，月と号数の計算を間違えていました。９月号に掲載します。それから，応募者はいらしたのですが，現在のところまだ正解者はおりません。
ワ：なあ，ホームズ。ちょっと問題が意地悪すぎたんじゃないのかい？
ホ：いや，すまない。では，ここでちょっとヒントを差し上げよう。
「人は探しているものを見つけると，探すのをやめてしまう」
ワ：なんだい，そのヒントは。全然CGと関係ないじゃないか。
ホ：応募者はいるが，正解者はいない。わかる人にはわかるはずさ。
ということで，締め切り(７月30日)は近いですが，皆さんもがんばって挑戦してください。

AFTER REVIEW

ビデオゲームアンソロジーシリーズ第3弾として登場したのは，ひたすらの連射ゲー，スターフォース。昔の熱さがよみがえる完全移植が好評です。α，β，γ，δ……，あなたはどこまでいけたでしょうか。

スターフォース

▶「完全移植とはこのことをいうのだー」と思いました。いまひそかにCU-21HDを買おうかなあと考えている(縦画面で遊ぶため)。 熊谷　武志(24)岩手県

▶「ゼビウス」を文化系シューティングとするならば，「スターフォース」は体育会系シューティングだ！　指躍る踊れシューティン。 中島　民哉(22)埼玉県

▶ラリオスとジムダステギの同時ボーナス達成にもう一度挑戦だあ。 中島　謙二(23)大阪府

▶いまのシューティングでは味わえない地味なキャラの攻撃がなつかし～！　カンストしたときは泣いたよ～！ 新谷　恵次(18)北海道

▶連射がいい。バランスがいい。スコア(ボーナス)が熱い。 三浦　栄悦(25)秋田県

▶ディスプレイを横にしての縦画面が気に入った。 君島　一彦(19)栃木県

▶腕がしびれる。 笹間　康弘(23)大阪府

▶腕がちぎれる。 伊藤　剛(18)大阪府

▶アーケード版はBGMがよく聴き取れなくてくやしかった。難易度もそこそこで，飽きがこないところがよいのだ！ 岩瀬　貴代美(21)福岡県

▶とにかく単純なところがスカッとする。9年前のものとは思えないほど新鮮です。 石川　博基(20)神奈川県

▶「撃って，よける」これがすべてでよい，真のシューティング。 高橋　陽一(19)埼玉県

▶破壊しまくる爽快感がいい。 明石　智仙(20)神奈川県

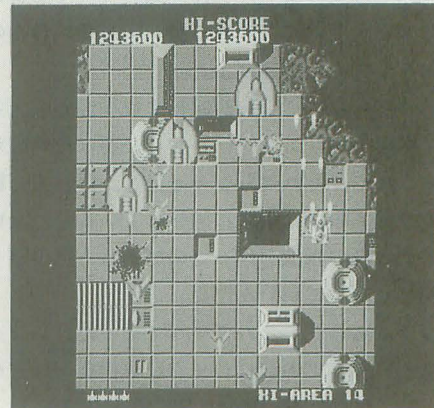
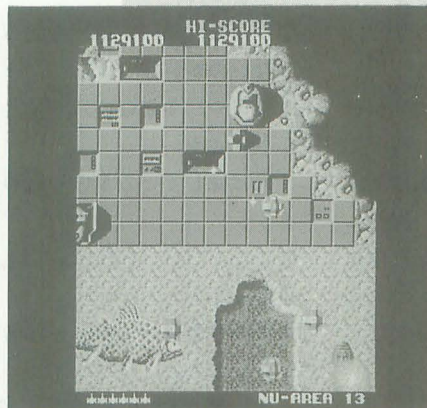
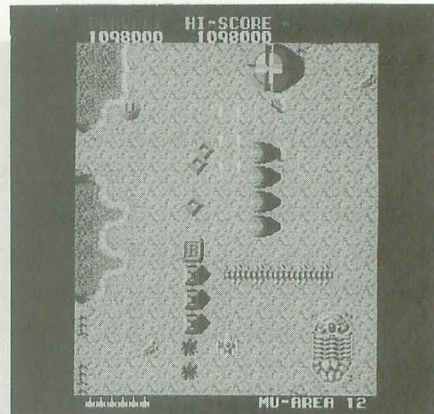
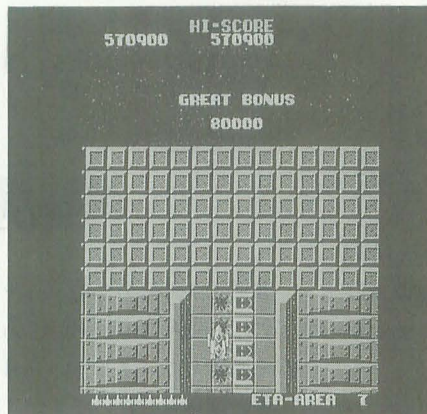
▶あの隠れキャラは私の夢でした。 久保田　智久(17)群馬県

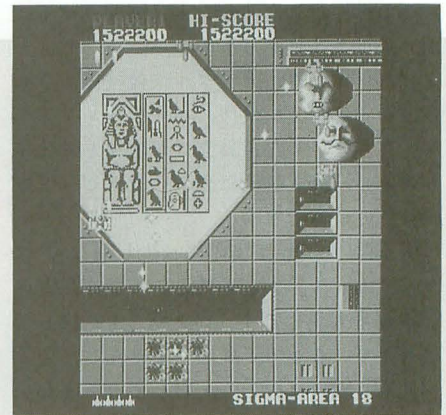
▶どこだゴーデス。 鈴木　勇(18)北海道

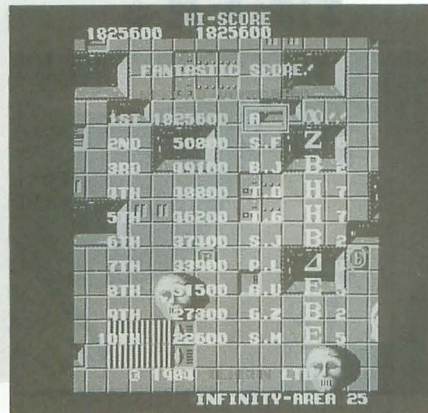
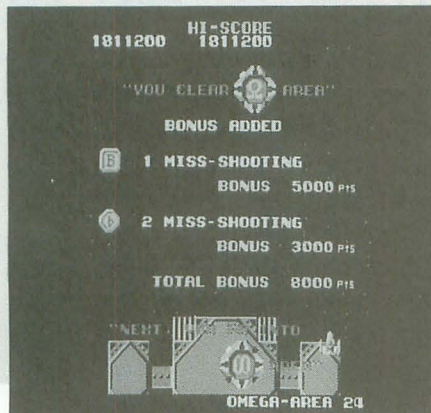
▶指がけいれん。ゴーデスこわれん。 小林　宏教(19)富山県

▶ボーナス加算のときαターゲットに「ぴったり」重なると，うれしい。 堂領　輝昌(19)神奈川県

▶一に連射，二に連射，三枝はオヨヨでぴりりと辛い。 界　洋士(24)大分県

▶ジムダステギボーナス×3が熱い。もちろんオート連射なし。

上田　考一(23)福岡県
▶これほど無心で楽しめるソフトはまずないぞ。価格も安いし，いうことなしだね。
越智　文昭(30)愛媛県
▶撃って撃って撃ちまくるのじゃー。何も考えることはない！　ひたすら撃つべし！
秋山　欣之(27)広島県
▶久しぶりに，ひと汗かいた。
畑中　英喜(19)大阪府
▶かつてファミコン版が出たとき「何か違う」とひっかかっていたわだかまりを取り去ってくれた。　田中　直樹(20)福岡県
▶やっぱり，弾よけできないのに難易度をDIFFICULT5にしてしまう麻薬的なゲームなのだった。　牧野　豊(20)北海道
▶これぞシューティングといったところ。紙一重でかわす快感。値段も気に入った理由のひとつ。　久保田　文彦(32)長野県
▶ジムダステギの近くがヤマだ。

黒畑　喜弘(19)新潟県
▶シューティングの本道で，８年たってもアツイ。　今枝　務(22)愛知県
▶ファミコンでは再現されなかったあの雰囲気を味わえる！　河野　太郎(20)東京都
▶単に懐かしいだけではない。それだけでは電波さんもわざわざ移植したりはしない。やはりいまあらためてやってみる価値があるのだ。俺もこのゲームを久しぶりにプレイし，失っていた何かを得たような気がしてならない。

あくまでピュアなのである。シューティングの真髄がここにあったのだ。現在多々あるこの系統のゲームがすっかり失っていた，根底に流れる熱い魂がここでは明らかに見いだせるのだ。小手先の技術もいらない。思考が行動を妨げることもない。ひたすらボタンを刻み，弾を見切ってよける。ただこれだけだが，その絶妙な，完成されたバランスの前ではこれこそ正道であり，これ以上は何も必要ないことだと納得できる。そう身体ができているのだ。

体感するのだ。もう，これはゲームではない。魂の奥に秘められた欲望を引きずり出すドラッグなのである。　(横内威至)

発売中のソフト

★銀河英雄伝説III　ブラザー工業(TAKERU)　6/20
X68000用　3.5/5″2HD版　7,800円(税込)
★リブルラブル　電波新聞社　6/25
X68000用　5″2HD版　7,900円(税別)

新作情報

★ザ・ワールド・オブ・X68000　電波新聞社　6/下
X68000用　5″2HD版　4,800円(税別)
★ロボットコンストラクションR.C.
エレクトリックシープ　6/下
X68000用　5″2HD版　8,800円(税別)
ブラザー工業(TAKERU)　価格未定
(3.5″2HD版はTAKERUのみ)
★宝魔ハンターライム#1
ブラザー工業(TAKERU)　7/9
X68000用　3.5/5″2HD版　1,500円(税込)
★宝魔ハンターライム#2
ブラザー工業(TAKERU)　8/6
X68000用　3.5/5″2HD版　1,500円(税込)
★餓狼伝説　魔法株式会社　7/23
X68000用　5″2HD版　8,800円(税別)
★悪魔城ドラキュラ　コナミ　7/23
X68000用　5″2HD版　9,800円(税別)
★ダーク・オデッセイ　ソフトプラン　7/下
X68000用　5″2HD版　価格未定
★芸夢じゃん　G.A.M　7/未
X68000用　3.5/5″2HD版　7,800円(税別)
★神々の大地〜古事記外伝　光栄　8/未
X68000用　5″2HD版　9,800円(税別)
★ギャラクシーシェイカーズ
ブラザー工業(TAKERU)
X68000用　価格未定
★コットン　EAビクター　9/24
X68000用　5″2HD版　9,800円(予価)
★ロボスポーツ　イマジニア
X68000用　5″2HD版　価格未定
★Traum　象スタジオ
X68000用　5″2HD版　価格未定
★鮫！鮫！鮫！　KANEKO
X68000用　5″2HD版　価格未定
★達人　KANEKO
X68000用　5″2HD版　価格未定
★エアバスター　KANEKO
X68000用　5″2HD版　価格未定
★サバッシュII　ポプコムソフト/グローディア
X68000用　5″2HD版　価格未定
★マージャンクエスト　SPS
X68000用　5″2HD版　価格未定
★麻雀悟空・天竺への道　シャノアール
X68000用　5″2HD版　9,800円(税別)
★クレイジークライマー1＋2(仮称)電波新聞社
X68000用　5″2HD版　価格未定

古くはトクサツ，最近はSFXといわれる架空の世界をスクリーン上に展開させる技術は，まさに日進月歩の勢い。コンピュータグラフィック，デジタル映像技術，さらにはハイビジョン映像，バーチャルリアリティなどといった道具立てが次々と進化することで，SFX技術は恐るべきスピードで成長しているのです。

古くは「2001年宇宙の旅」，少し前なら「スターウオーズ」シリーズといった作品群のSFX技術も見事です。いま見ても，決して色あせているものではありませんし，とくに日本映画に真似しろといってもできないでしょう。「バロン」しかり，「ターミネーター」「トータルリコール」しかり。挙げていくときりがないので，これくらいにしますが，ハリウッドSFX映画の名作は本当に枚挙にいとまがありません。

ただ，こうした優れたSFX映画群とはまったく次元が異なるSFX映画が登場するとは，正直いって思いもよりませんでした。これが凡人の悲しいところで，「よりよいもの」を発想するとき，どうしても従来の延長線上をトレースしようとしてしまうのです。だから新しい次元を開拓することはできないのでしょう。

ちょうどこの号が発売される頃に公開される１本の映画は，まさしく過去からの既成概念をいとも簡単に打ち砕くSFX作品だったのです。さすが天才といわれる人物が送り出した作品ならでは，というところでしょうか。

映画の内容は，ある富豪がバイオ技術を駆使して，ある小島で6500万年前に絶滅したはずの恐竜を再現させてしまうという物語。この島で起こるバラ色の夢と，絶望的なヒューマンエラーをめぐるSFサスペンスドラマです。

内容もサスペンス度も文句なしに二重丸なのですが，とにもかくにもこの映画の売り物は「新世代SFX」とでも呼ぶべき，あまりにもリアルな非現実映像です。最初に恐竜が出てきたシーンでは，愕然とさせられてしまいました。

これまでのSFX映画の名作は，いかに優れたSFXであっても，見る側に「これは作り物の映像なのだな，フムフム」なんていう表現しづらい一種の安心感を与えてきたことは見逃せません。しかしこの作品が見せてくれる映像は，実際にどこかに生息している恐竜を使って撮影しているのではないか，と思わせてしまうほど驚異的です。

この映像を支えるのが最新最高級のデジタル画像処理技術だとか。同じ売り文句であっても，UFOキャッチャーに入っていそうなモロぬいぐるみの蛾の怪物が出てくる映画とはわけが違うことは，一目瞭然。妥協なき現実感とでもいうべきリアリティをストレートに目の中に押し込んでくるかのようです。

スティーブン・スピルバーグ監督最高の作品，というだけの内容不明の前宣伝や拡大公開態勢が奏功してか，６月11日のアメリカ公開時には，わずか９日間で興行収入

X-OVER・NIGHT

（クロスオーバーナイト）

［第37話］

人間は電気羊の幻影を見る

TAKAHARA HIDEKI 高原 秀己

１億ドル突破という途方もない興行成績をあげている映画「ジュラシック・パーク」のお話を延々と書いてきました。

デジタル画像処理技術が飛躍的に進歩したからこそできたことで，過去には真似のしようのないことです。まさにエレクトロニクス様々。映画という，娯楽であり芸術である分野の表現方法に完全に新しい１ページを切り開いたのです。そして，この先ますます高度になっていくことは間違いありません。

ですが，このような作品を目の当たりにしてしまうと，SFXという存在自体が何かしら怖くなってきます。どこまでが現実の道具立てで，どこまでが作り物なのかという区別がつかなくなってしまうわけで，今後は確実に非現実が現実に自由にミックスされてくることを意味しています。

極端な話が，見たことのない絶世の新人美人女優が，実はデジタル映像の産物である架空の存在だった，なんてことも容易に起こりうる時代であるのです。幻の古代文明が生き続ける村が発見された，などというニセモノの映像だって，簡単にできてしまうんですから。

この心配は，いうまでもなく受け手の許容力とも密接にリンクします。さてここで見逃せないのが，おとなにも大人気という例の子供番組。瞬間芸ノリの短いコーナー多数をはさんで，CGによるさまざまなキャラクターが次々と登場，主演の少年少女とおしゃべりをします。CGが児童画っぽいので，おとなが見ると，たいしたことはないのですが，見ている子供にとっては，シュール君やらプラネットちゃんは何らかの意味で「実在」するという感覚をもっているはずです。

このフジテレビ「ウゴウゴルーガ」の大ヒットに，あわてて類似番組を作るテレビ局まで出てきています。子供たちはCGによる架空のキャラを受け入れる素地をいつしか身に備えてしまっているわけです。

この子供たちが育っていくのですから，「ジュラシック・パーク」で幕を開けるであろう「新世代SFX」の映像に対する警戒感などは完全な杞憂に終わってしまうことでしょう。

まあ，こうした不要な心配をしてしまうこと自体が，ぼくなどはトシなのかもしれないのですね。

しかし……。

みかん星人やトマトちゃんがバイオ技術で実在するキャラとなり，逆にウゴウゴ君やルーガちゃんはデジタル映像の産物である，なんていう時代もやってくるかもしれないわけですね。そのときはオープニング映像の居酒屋の場面で，おじさんが食べている料理(気がついていない人は録画してスロー再生して見てみるように！)が絵空事ではなくなるかもしれません。

で，何はともあれ。

ああ，いけない。

「ジュラシック・パーク」を見終わってからもう６時間がたつのに，まだドキドキしています。

illustration：Haruhisa Yamada

愛読者プレゼント

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1993年8月18日の到着分までとします。当選者の発表は1993年10月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号の他の懸賞には当選できない場合がありますのでご了承ください。

ファミリーソフト
☎03(3924)5727

ヴェルスナーグ戦乱

X68000用　5"2HD版
9,800円(税別)　3名

フルマウスオペレーションで自動戦闘のお手軽ロールプレイングゲーム。

②
ブラザー工業
☎052(824)2493

幻影都市

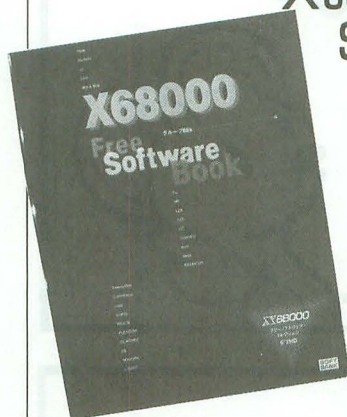
X68000用　3.5/5"2HD版
6,800円(税込)　3名

キャラの動きがリアルな近未来ハードボイルドアドベンチャーゲーム。

ソフトバンク
☎03(5642)8100

X68000 Free Software Book

2,900円(税込)　5名

プログラムの解説，入手方法，座談会など，フリーウェアにまつわる話題を収めた書籍。フリーウェアを収録したディスク付き。

ソフトバンク
☎03(5642)8100

X680x0 libc

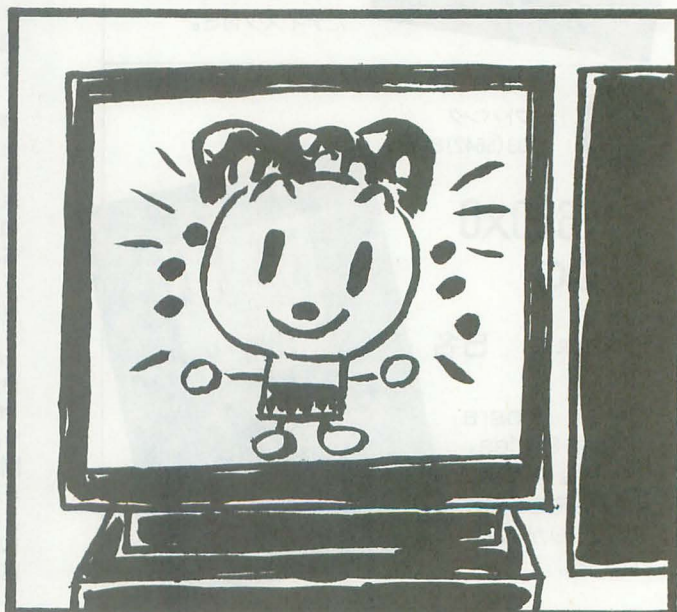
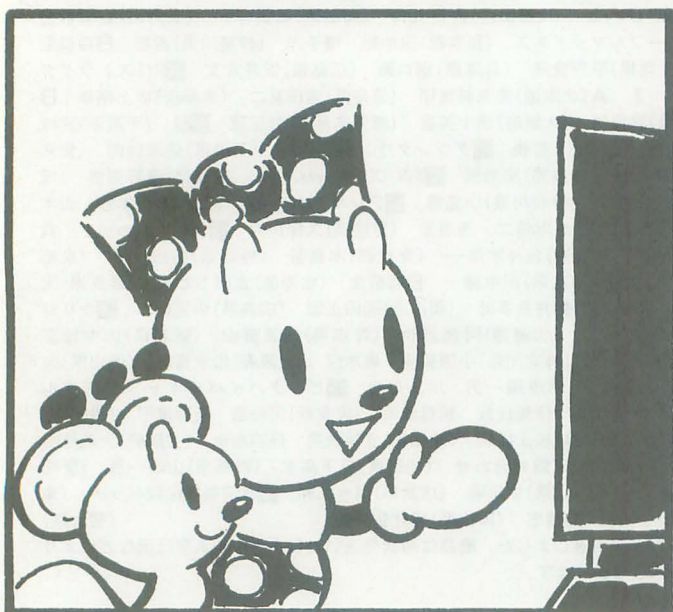
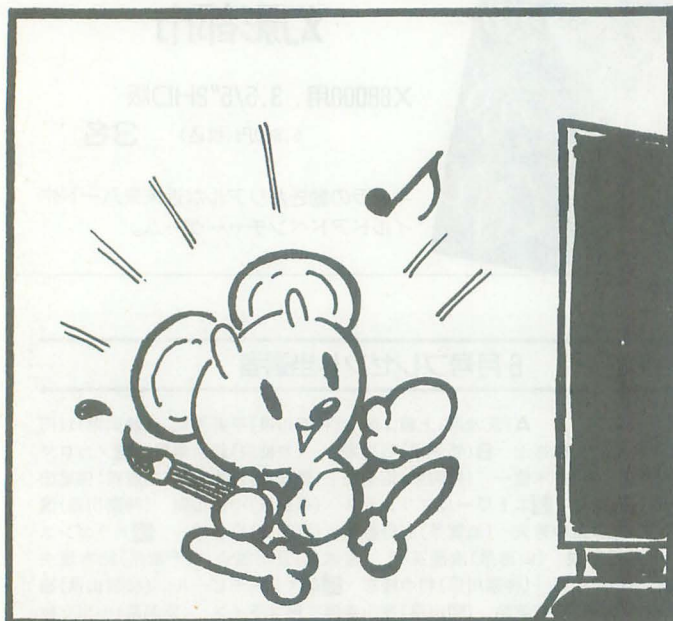
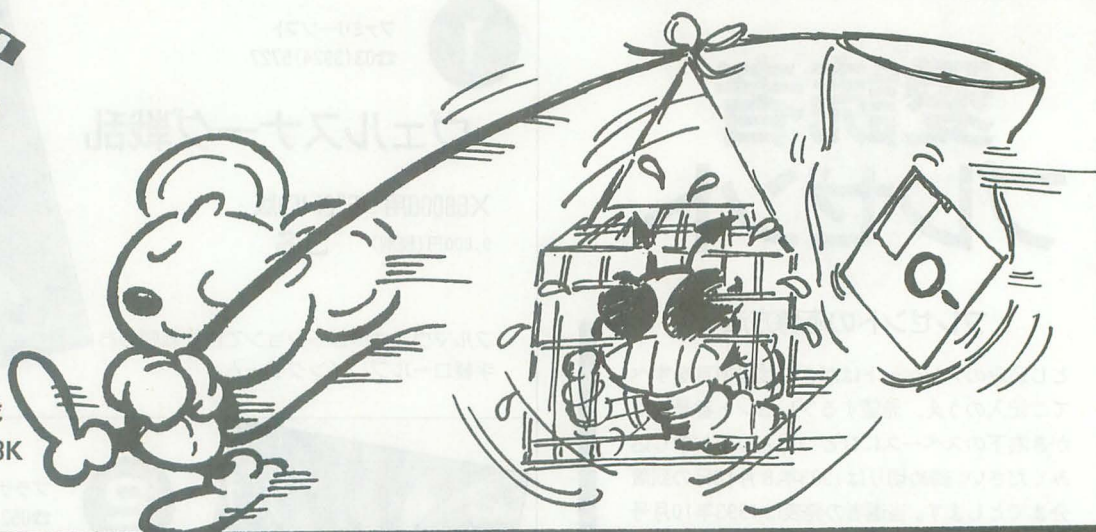
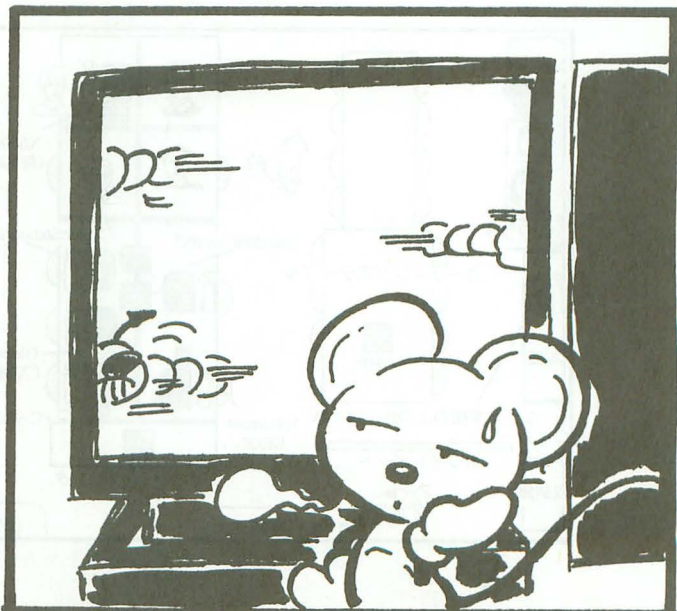
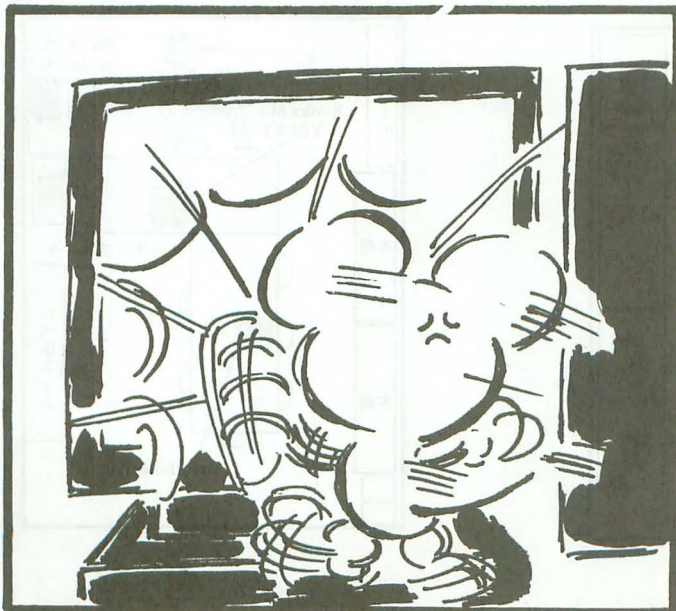
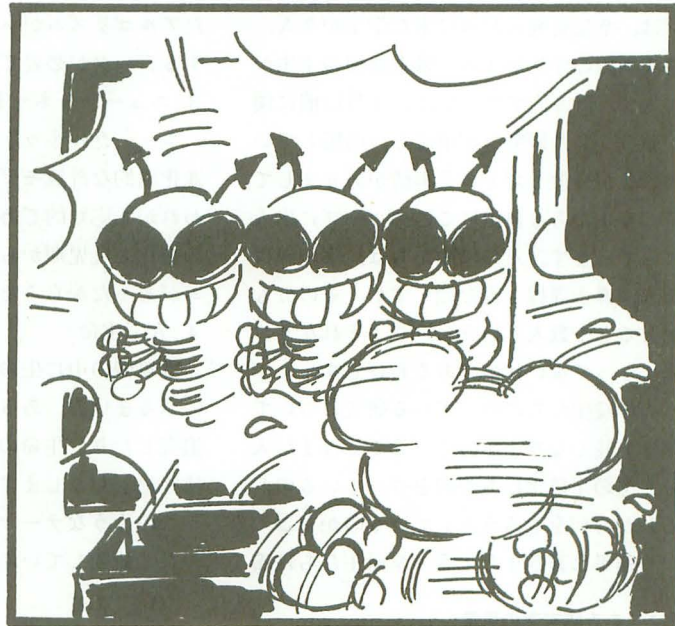
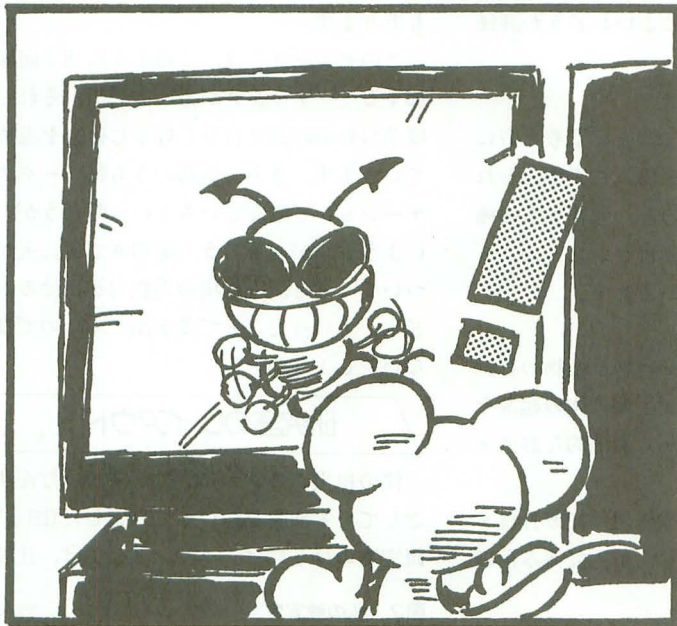
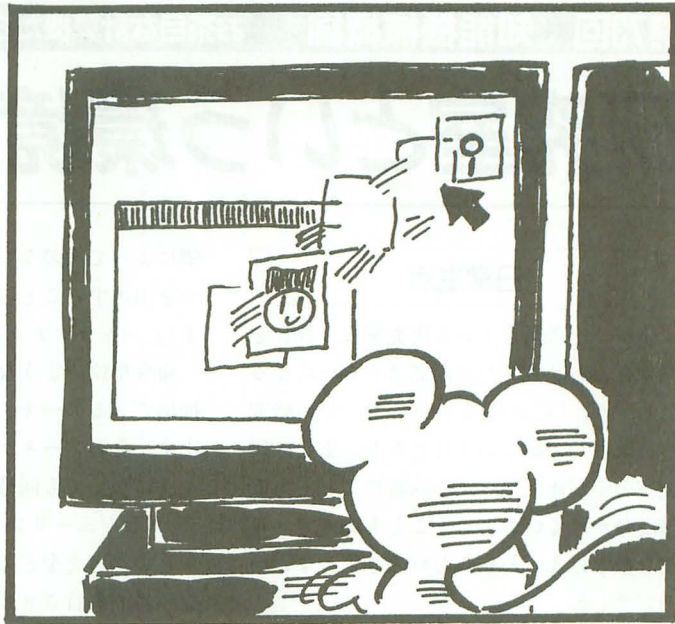
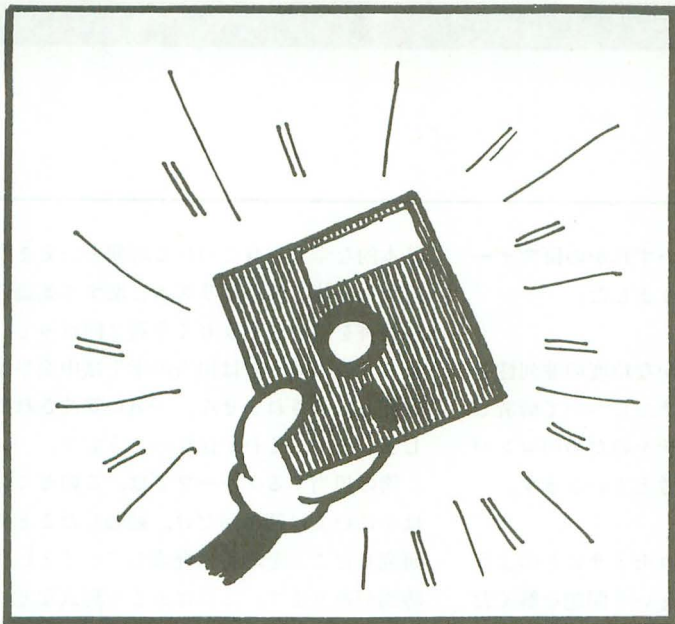
7,800円(税込)　5名

「X68k Programming Series」の第2弾。XC，GCCで使用できる関数ライブラリが収録されている。

6月号プレゼント当選者

1スピンディジーⅡ　A(茨城県)上籠洋昭　(神奈川県)平井秀司　(静岡県)坂田清明　(愛知県)大橋修治　B(新潟県)石川栄一　(京都府)飯村幸男　2メガロマニア　(神奈川県)古木健一　(静岡県)沼圭司　(愛知県)八木薫　(鳥取県)堀尾忠教　(愛媛県)東寛　3エトワールプリンセス　(福島県)小桧山剛　(神奈川県)清水勲　(富山県)指中芳夫　(滋賀県)志野敏彦　(京都府)平谷淳一　4ドラゴンスレイヤー英雄伝説　(山形県)渡邊英行　(宮城県)及川雄也　(千葉県)鈴木康夫　(東京都)田中剛一郎　(神奈川県)村中隆志　5Aスターモビール　(和歌山県)福田雅彦　(兵庫県)敏森健裕　(岡山県)遠山幸男　Bスライス　(新潟県)小川比佐夫　(東京都)辻康介　(富山県)杉林隆志　6銀河英雄伝説Ⅲ　(北海道)石田伸吾　(埼玉県)鈴木真一　(富山県)吉村元伸　(鳥取県)石賀伸一　(広島県)野瀬哲宏　7Aマーブルマッドネス　(東京都)田中聡　増子洋　(神奈川県)西惹　B将棋聖天　(千葉県)平野貴隆　(兵庫県)樋口崇　(広島県)安井常文　8F15ストライクイーグルⅡ　A(北海道)東海林寛昭　(福井県)吉田真二　(京都府)坂上裕幸　B(群馬県)武藤勝　(大阪府)畑中英喜　(鹿児島県)新村正蔵　9極　(千葉県)宮城照彦　(兵庫県)松下哲也　10ダウンタウン熱血物語　(石川県)佐渡詩郎　(愛知県)池田伸治　(奈良県)柳瀬薫　11XIN/OUTⅡ ver.7.0f　(宮城県)浅野克博　(埼玉県)松木明博　(神奈川県)小島修　12テレホンカード　(東京都)金本修　山本知治　(神奈川県)岩本修二　浅原元　(愛媛県)久保民秀　13テレホンカード　A雀JAKA雀　(北海道)佐々木淳一　(東京都)木島智　(静岡県)藤田康一　(京都府)西村武雄　(奈良県)田中謙一　B同級生　(北海道)太田もとき　(群馬県)天海宏人　(東京都)櫻井良多郎　(岡山県)岡田正宏　(広島県)中光雄二　14オリジナルボールペン　(北海道)阿部達也　(青森県)三浦直樹　(新潟県)山中雅彦　(東京都)広野徹　(神奈川県)小原英隆　黒木恒　(大阪府)松永貴輝　(岡山県)大野敏郎　(福岡県)伊規須一男　川上卓也　15ビックバイパーキャストモデル　(埼玉県)新井和夫　伊藤民哉　諸星城治　(東京都)河野浩　近澤淳平　(長野県)竹前和哉　(富山県)松永好司　(愛知県)立原秋男　丹羽浩也　(大阪府)大森啓明　16オリジナルグッズ詰め合わせ　(埼玉県)日下高志　(群馬県)山崎一馬　(愛知県)岩田剛　(和歌山県)中田聡　(広島県)清水弘和　17銀河英雄伝説バッチ　(東京都)田中和也　前橋忠　(岡山県)藤井弘樹　(敬称略)

以上の方々が当選しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

another cg world
今回のCGデータ
キャラクタ1体の
物体数は63
合成で処理
マッピングデータ作成
Z's STAFF PRO-68K

研究室という環境

日常生活

僕がいま所属している研究室は，教官と学生を合わせるとたぶん35名ぐらいになるでしょう。「たぶん」というのも，この研究室に入ったのがこの4月であり，また，研究室の中が研究テーマや教官によってある程度分かれているということもあって，正確な数字はよくわからないのです。すごい数ですしね。

この研究室の中で，僕が担当するグループは，卒業研究のために来た学生が7人，修士課程の学生が1人，博士課程の学生が1人で，計9人です。でも，4月以前に僕が属していた研究室が消滅し，所属していた学生が分散したという事情がありまして，その学生たちも自由にこのグループに出入りしています。その学生たちは，修士課程2人，博士課程1人なのですが，その分も加えると十数人というわけで，それなりに僕にとって重い仕事であるわけです。

大学院生たちのやっている研究について書くと長くなりますので，ここではまだ入りたての卒研生たちが何をやっているのか，これから何をやろうとしているのかをごくごく簡単に書きましょう。卒研生たちは希望によって次のうちのいずれかの研究テーマを担当することになりました。

1.アーキテクチャ

命令単位のような細かな粒度の並列性を抽出できるアーキテクチャについて研究します。そのアーキテクチャのためのコンパイラについても同時に考えていきます。

2.スケジューリング

どのタスクをどのプロセッサにどのような順番で実行させるかという問題を解くためのアルゴリズムを考え出します。提案したアルゴリズムが本当によいかどうか評価するデータが必要です。

3.ニューラルネット

ニューラルネットの学習という考え方に進化論的な計算モデルを導入します。われわれが一応知的であるのも，大昔からの種の進化と乳児期からの学習というメカニズムがあったからこそといえます。

4.人工生命

計算機の中に生命体を育て，生物の行動のふるまい方，あるいは生命自体の起源を追究したり，生命の中から論理的な形式を抜き出したりします。

このようなテーマを卒研生は10カ月ほどかけて研究していくのですが，夏休み前は基本的な学力を身につける時期といえます。研究室全体でアルゴリズムに関する英語の教科書を読むというゼミを週2回行っています。このゼミでは担当の学生は中途半端な発表は許されません。一気に撃沈されてしまいます。それが伝統のようです。

僕の担当するグループでは，このゼミのほかにいまは週1回だけ，勉強したことや研究したことを順番に発表していくという時間があります。これはあまり形式などにはこだわらずに皆が自由に議論できる場でもあります。

このグループでは，このように週3回参加するミーティングがありますが，それと授業以外の時間は自分で好きに研究を進めていきます。まあ，最初のうちはワークステーションで遊んでいるといったほうがいいようですが，そのうち締切りで尻に火がついてくると，火事場の馬鹿力を出せるかどうかといったことで差が出てくるのではないでしょうか。

研究室のレイアウト

僕の担当するグループの学生がふだんすごしている部屋を図にしてみました(図1)。研究室のほうには，マシンとしては，比較

図1　ある研究室の風景

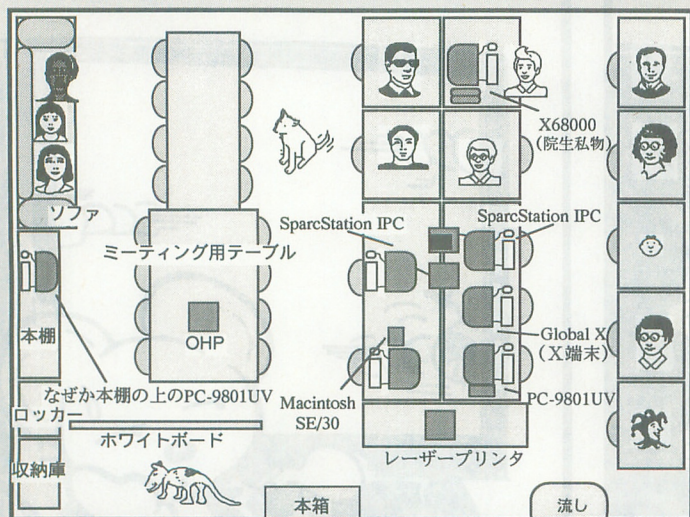



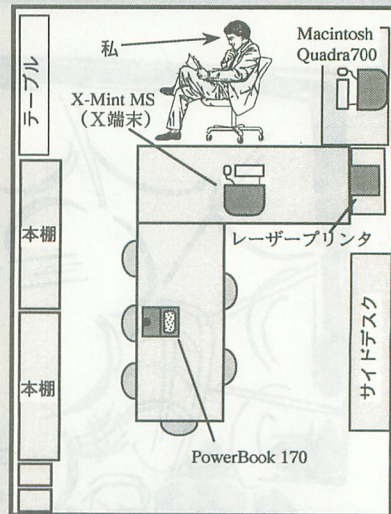
図2　私の教官室

的安いワークステーション2台，X端末1台，パソコン（X68000，Macintosh，PC-9801）があります。僕としては，9人が過ごす環境としては最小限度ぐらいかなと思っています。そのうち取り合いが起こるかもしれません。

図2は僕が毎日過ごしている部屋のレイアウトです。X端末は最近入ったばかりです（代わりにワークステーションが出ていってしまいましたが)。図1の部屋のSparc Stationをホストマシンとしています。

僕自身はまあまあ使いやすい部屋だなと思っています。参考までに，教官室のレイアウトのいくつかの典型例を示します。図3は比較的よく見るタイプです。自分の勉強のためのスペースと外来者や学生との打ち合わせのスペースをはっきり切り分けるのが特徴です。図4のような社長室レイアウトも比較的よく目にします。机が入り口を向いているので迫力があります。お客さんとの打ち合わせはソファで落ち着いてできます。図5は机には秘書の女性がいて手前のテーブルに教授が座るという例です。狭い部屋の場合には合理的なのでしょうか。

文系の研究室はあまり詳しくないのですが，本があふれかえっていて，このようなはっきりしたレイアウトさえ書くのが不可能な例を僕は知っています。教官が座る場所以外はすべて本なのですから。

さまざまな研究室

大学だけに限っても世の中には無数の研究室があります。僕自身もいくつかの研究室に加わっていましたし，数多くの研究室について知っているつもりです。とにかくさまざまな個性を持つ研究室がありますね。

たとえば，遅刻したら罰金を払わねばならないというルールを作って学生たちを朝早くから来させる研究室もあれば，夜になったらゾロゾロと出てくるような学生ばかりの研究室もあります。いやそれどころかまるで無人の研究室もあります。あるいは，裕福にお金を使えるところもあれば，たいへん貧乏な研究室もあります。教授が神様のような研究室もあれば，ひどく腰の低い教授の研究室もあります。

想像を絶するような研究室もありましたが，結局，いまのところ，研究室の運営方針とかポリシーとか雰囲気とかいうものが，アウトプット(学生に対する教育効果とか研究業績とかいうもの)に対してどのような影響を与えるのかという大事な問題に関しては，残念ながらこれといった結論を僕自身得られていません（そもそもないのでしょうか)。

たとえば，学生に対して教育というものをあまり行っていないように見える研究室でも，学生は先生がたの研究や研究に対する姿勢をきちんと見ているように思えますし，逆にきちんとした教育をして成果をバシバシあげているような研究室でも，意外と学生は隙を盗んでサボることばかり学んでいるという面もあるような気がするのです。

ただ，どこの研究室についてもだいたいいえるのではないかということがひとつあります。それは，その研究室の教授のタイプと所属している学生のタイプはよく観察するとすごく相関があるということです。

これは，似たような研究テーマに興味を持つ人間は性格なども似てくるとか，学生は教授の性格になじめそうなところを無意識的に希望しているとかいう理由で説明されるかもしれませんが，いずれにせよ，教官があれこれ学生にいう分だけ自分にも直接的にはねかえってくるものですから，うかうかとした人生をぬけぬけと送ることは難しいのですね。

図3　典型的な空間分割型教官室

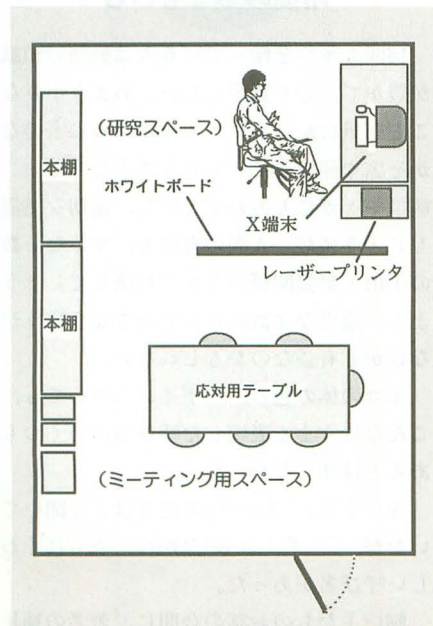



図4　典型的な社長室風教官室

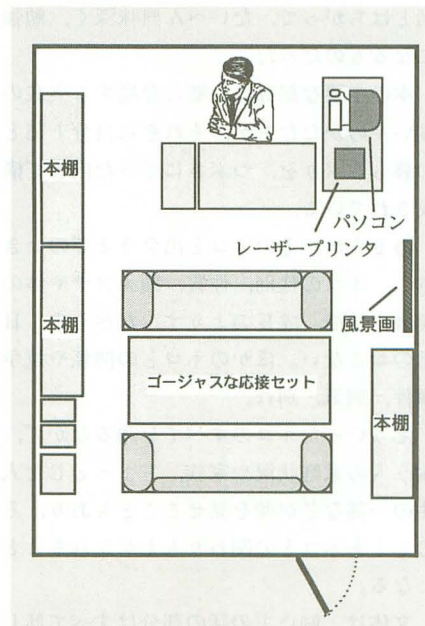
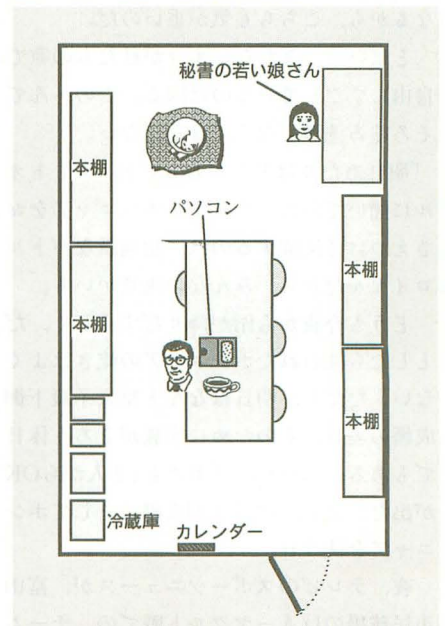



図5　狭い場合の合理的教官室

ネコを洗ってVJE-β

Takazawa Kyoko
高沢 恭子

季節の変わり目は，気分一新リフレッシュにもちょうどいい機会ですね。キョウコさんちで今年の衣がえのターゲットになったのは，ホンニャアと辞書から飛び出す例のカエル君。さて，その成果は？

降っても晴れても美しく，くもり空ならおだやかでやさしい。じつは，じわじわと暑さがしのびよる季節だけれど，6月には若々しさと落ちつきがある。

向かいの中学校の生徒たちも，高校生のトオルも，制服が夏姿になった。衣がえは学校生活の古くからの慣習なのに，ハッとするようなすがすがしい演出だ。その白さ，軽快さへの変身で活気もわいてくることだろう。

ホンニャアの衣がえは，随時，すこしずつおこなわれているらしいのだが，彼の毛皮の衛生状態はたぶんよくない。

あいかわらずの入浴ギライで，洗ってやるとなると，家族3人総がかりの大行事になるから，こちらも気が重いのだ。

とはいえ，きたないネコが私たちの家で自由にすごしているのは困る。このへんでそろそろ洗浄しなくてはならない。

「明日あたりはどうかしら」と，夫とトオルに聞いてみた。抵抗するホンニャアをおさえつけて洗濯するのは，超過激なバトルロイヤルだから，みんなの決意がいる。

どうも今夜から雨が降りだすらしく，だとしたら洗われたホンニャアの乾きはよくない。ただし，明日はなんと皇太子殿下御成婚の当日，そのために家族がそろう休日でもある。「いいよ」「ボクも」2人からOKが出た。よし，明日は御成婚，そしてホンニャアを洗う日。

夜，テレビのスポーツニュースが，富山市民球場の巨人ーヤクルト戦での，チーム同士の大乱闘を伝えていた。5月以来，死球つづきの因縁が尾をひいて，おたがい闘志がむきだしになっているらしい。

「ここも暴れてるね」

明日のわが家を想像して，3人で笑った。

ネコを語る人たち

新沢ひろ子さんの『お風呂が好きなネコもいる』という本をもとめたのは，やはりタイトルにひかれてのことだった。

ネコはお風呂がキライであたりまえ，ホンニャアをとくに批判してはいけないらしい。でも，できればこれを読んで聞かせたいなんて。

ところがこの本は，楽しいだけのネコの話とはちがって，たいへん興味深く，勉強になるものだった。

本の主要な部分は，順に登場する実在の飼い主のかたたちが，それぞれ自分と猫との暮らしぶりを，つぶさに語った内容で構成されている。

ある人が，そのネコと出会うまでのいきさつ。ネコの性別，性質，顔カタチや体の美醜，特徴。成長のようす，たべもの，日常のおこない。ほかのネコとの関係や抗争，事件，病気，別れ。

そういったネコのすべてを語るなかで，飼う人の家庭状況や家族，ちょっとした人生の一部などが顔を見せることもあり，そのこととネコとの関わりもえがかれることになる。

文体は，飼い主の話の部分はすべて話し言葉で，おそらくその人の言葉づかいそのままを，なるべく修正しないつもりで再現したと思われる。そのために，登場するネコたちの生活が，飼い主もふくめて，まるごとリアルに目に浮かんでくる。

たくさんのネコの話を聞かせてもらいながら，読みすすむにつれて，いくつものことを考えた。

ネコが人間の言葉や気持ちをよく理解し，こまやかで頭脳的な反応を見せるというのは，もうよくわかった。

それよりも，もっと感じ入ってしまったのは，飼っている人たちがネコと真剣に接していることだった。どの人もネコをあなどっていないし，甘やかしてもいない。ネコを守ってやりながら，ネコとのきまりをきちんとつくり，おたがいの快適を考えている。そのうえでネコのいる生活をじゅうぶん楽しんでいる。

多くの人が，ネコを1匹でなく複数で飼っていることにも感心した。そして，1匹しか飼わないことが，いかにネコ観を貧しくさせているかを知った。

ネコは2匹，3匹という関係のなかで，いちだんとおもしろく個性を発揮していくようだ。それが，親子や兄弟の場合もあり，まったく種類のちがう同士だったり，または犬であったりもするが，その複数のなかでの対立や助けあいがほんとうにおもしろい。人間のなかにたった1匹で暮らす動物には，その面が見えてこないのだ。

暗闇のネコもいる

何匹もネコを飼っている人は動物の知識が豊かで，心も寛容らしい。あまり小さなことを気にかけずに，ネコのために家のなかを広く開放している人もすくなくない。病気やケガにもあわてないし，適切な処置もじょうずだ。人間の育児も，すくない数の子供と緊張関係のなかで対決してしまうより，適当な多数のなかでおこなうほうが，なにかと有益なのかもしれない。

ネコの体の色彩，デザインについても，こんなにうまく表現した呼び名がいくつもあるとは知らなかった。

「キジネコ」「茶トラ」あたりはよく聞いていたが，「三毛ネコ」のなかに，さらにくわしい呼び名があった。

飼い主たちのお話の合間に，著者の補足

や所感がつづられていて，いろいろなことを教えていただいた。

日本の伝統的な三毛ネコは，白地に茶色と黒がブチになっている。ところがそれがゴチャゴチャになってあまりきれいでない場合は，三毛でも「ガチャ三毛」という。これで白がなくなると「べっこうネコ」といい，英語では「トーティシェル」というそうだ。

tortoise-shell＝「べっこうの」と辞書にあった。このごろは洋ネコとの混血がすすんだせいか，べっこうネコがふえたと新沢さんはおっしゃる。「三毛じゃなくて2色なんで『ニケ』なんですよ」といっていたFBI-NETの「ちゃがま」氏の飼いネコ，ニケちゃんは，べっこうネコにちがいない。

ネコの寿命についても，ホンニャアがいつまで元気でいるだろうかと考えるから，興味のあることだ。

日本で記録に残る最長寿のネコは，36歳半で，2番目は28歳だそうだ。つぎは26歳半，どうもメスネコが多いようだ。本のなかにも，20歳くらいのネコが何匹か登場する。

ネコの年齢を人間の年齢に換算する方法もいくつかあるらしく，私もいろいろな説を耳にしたけれど，この本に紹介されている計算法は，いちばん無理がないように思えた。

まず1年で18歳，2年で24歳，そのあとは1年ごとに4歳をくわえていくという方法である。鉄道やタクシー料金の計算法みたいだが，成長までが短時間で，あとはゆるやかに老いていくらしい考えかたはなっとくができる。ただし，この方法で換算すると，ホンニャアは10歳近いので，50歳をすぎてしまうことになる。

もうひとつ，この本には，激辛の香辛料が入っている。

愛されながら，気ままに元気いっぱい，人間と暮らしているネコ。狩猟を楽しんだり，いっしょにバスタブにつかったりするネコたちの話といっしょに，陰の世界で虐待されているネコの実態が，ドキュメントで語られているのだ。

飢えさせ，瀕死に近い状態にした動物たちを，同情させることで売る露店商。ペットショップの秘密。三味線用のネコをアルバイトを使ってあつめる業者。

ペットがもてはやされて，動物といえばほほえましいという一面だけが伝えられることが多いなかで，やっぱりこれも，知らなければいけない話だと思う。

6月9日，皇太子殿下御成婚の日に，ホンニャアはめでたく猫用シャンプーで洗いきよめられた。本人がとても神妙な態度で協力的にのぞんだので，今回のケガ人はゼロ，そのあと雨もあがった。

illustration：Kyoko Takazawa

カエルと別れるために

わが家のPC-9801VMで，9年近い功労のあった日本語FEP，あのカエルの出るVJE-Σは，名誉をたたえて殿堂入りとなった。代わって当面の救援投手はVJE-β。

Σからβになって，内容や機能で画期的な進化があったかというと，答えは「あまりない」。ただ，気分が変わったために，前よりカエルにこだわらなくなった。

つまり，「帰る」「返る」「代える」などが，みんな「蛙」になる心配を，気にかけなくなったのだ。もし，またそうなったとしても，これでアキラメがつく。

利点はこれだけ。ところが，思わぬ問題がついてきた。キー操作が前とちがうということ。それもわずかにちがうという小さな変化が，やっかいな波紋をよんだ。

これには，ビックリ。

Σの場合は，漢字変換したあと，スペースキーで確定となって，文中に置いていく操作方法だった。それが，βではNFERキーで確定する。変わったことといえば，基本的にはこれくらいだ。

自分が理解すれば，ただ叩くキーを換えるだけ，ただそれだけのことだ。なのにそれはたやすくなかった。なにしろ9年間，指がおぼえてしまった動きはシブトイ。いくら確定のたびにNFERキーを叩こうとしても，指はスペースキーを叩く。

自分の意思ではない。指はかってにひとりで動く。独立して踊る。せっかくカエルコンプレックスが遠のいたと思ったのに，指がカエルのようにはねている。

長さのあるスペースキーは，左右どちらの親指でも使えた。NFERキーを打つとき，じっさいはどの指を使うのがノーマルなのか知らないけれど，私は左手の親指が出る。いままでは，そのときどきのタイミングで，2つの親指のどちらかでラクに打っていたのに，こんどは左だけにきめられて，それも，ちょっと内側に指を入れるかたちで打たなければならない。2倍の便利さが，半分以下になった。

いままで苦労知らずだった左の親指が積極的にはたらくことを強いられて，これが左手全体の，さらにはそれにつられた両手全部のトラブルになった。ふだん，キーを見て打つことはしないから，指がおぼえていたキー同士の距離感が狂うと，まともな文章は画面に並ばなくなる。

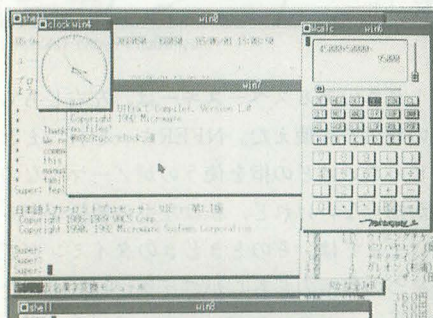
こんなことなら，Σのままのほうがよかった，なんども後悔した。また，カエル君が古池からとびだしてもいい。しかし，あと戻りとはナサケない。珍妙な漢字変換，文節変換もすくなくなったように思うし，全体的にはやはりβはΣにまさる内容だと認められる。すこしでもそう感じたら，劣るほうに戻るのはさけたい。

そう思っても，ちょっと気をぬくと指がΣ時代の一連の動きをはじめる。9年間は長かったのだ。ひとつの段階を越えようとしたら，多少ともショックとストレスにたえなければならないと考えて，いま努力のさいちゅうである。

『お風呂が好きなネコもいる』新沢ひろ子（にいざわひろこ）著　草思社刊

PENGUIN INFORMATION CORNER
ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

X68030対応マルチタスクOS
OS-9/X68030 V.2.4.5
マイクロウェアシステムズ

OS-9/X68030 V.2.4.5

マイクロウェアシステムズはX68030対応のマルチタスクOS「OS-9/X68030 V.2.4.5」を発売した。

「OS-9/X68030」はマイクロウェアシステムズが開発した，モトローラ68系対応のリアルタイムOSで，マルチタスク，マルチユーザー環境を特徴としている。また，自由にユーザーの仕様に合わせてシステムモジュールが選択できるモジュール構造も大きな特徴である。

X68030版でも，このOS-9の特徴である，複数のタスクを同時に実行できるマルチタスク，複数のアプリケーションを同時に多人数のユーザーが利用できるマルチユーザー環境をサポート。X68000版と同様にパーソナルウィンドウを搭載し，よりマルチタスク環境を快適に利用できるようになっている。

もちろんSCSIハードディスクもサポートしている。これにより，大容量ハードディスク，光磁気ディスクなどの接続が可能となった。また，日本語FEPとしてVJE-γ V2.0を採用し，日本語環境の整備が図られている。

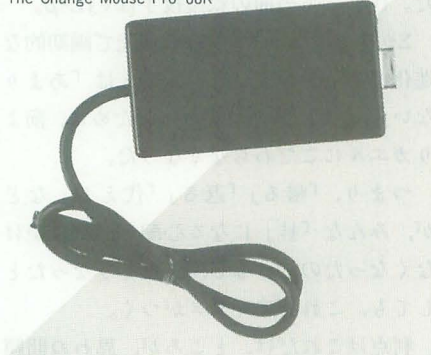
パッケージには，kernel,scf,rbf，パーソナルウィンドウ，プリンタドライバ，RS-232Cドライバ，SCSIハードディスクドライバ，FDドライバ，日本語FEP VJE-γV2.0，shell.ユーティリティコマンド，インストレーションマニュアル，オンラインマニュアルが同梱されている。

価格は25,000円(税別)である。

〈問い合わせ先〉

マイクロウェアシステムズ㈱　☎03(3257)9003

X68000/030用98バスマウスアダプタ
The Change Mouse Pro 68K
東京システムリサーチ

The Change Mouse Pro 68K

東京システムリサーチでは，X68000/030にPC-9801用のバスマウスを接続するためのアダプタ「The Change Mouse Pro 68K」を発売した。

アダプタをマウスとX68000/030の間に接続することで，PC-9801用のバスマウスを使用できるようにするためのもの。

基本的に使用できるマウスは，光学式エンコーダマウスタイプで，機械式エンコーダマウスの，完全な動作保証はしていない。

また，アダプタには本来のマウスのカウントを4倍速で動かすための「ハイスピード」と，自動的にカウントを切り替える「AUTOカウント」の切り替えスイッチがついている。

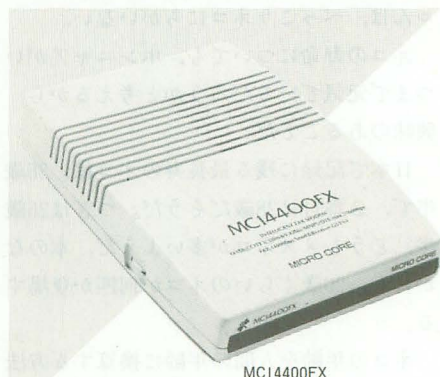
現在，以下の3機種について動作を確認済みである(Oh!X独自調査)。

・MK MOUSE　和知電子
・AM98 Auto　アーベル
・EGG MOUSE AV　エレコム

価格は9,800円（税別）となっている。

〈問い合わせ先〉

東京システムリサーチ㈱　☎0425(28)1824

14400bps対応のFAXモデム
MC14400FX
マイクロコア

MC14400FX

マイクロコアは，インテリジェント型FAXモデム「MC14400FX」を7月下旬から発売する。

「MC14400FX」は，通信，FAX通信とも最大速度14400bpsの転送速度をもつFAXモデムで，データ訂正機能CCITT V.42を搭載。データ圧縮機能はMNPクラス5，CCITT V.42bisに準拠，FAXはG3規格に対応している。

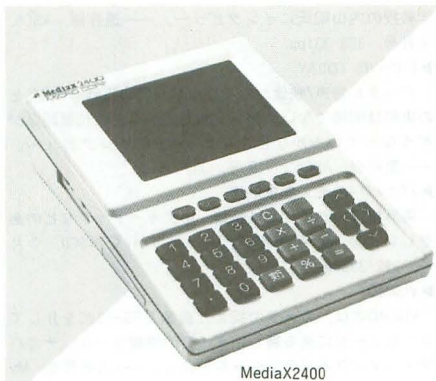
価格は，46,800円（税別）。

〈問い合わせ先〉

マイクロコア㈱　☎03(3448)0811

マルチネット対応型電卓
MediaX2400/1200
マイクロコア

マイクロコアは，マルチネット対応型電卓「MediaX2400/1200」を10月下旬より発売する。

両機とも，通信機能を備えた電卓であり，ディスプレイに320×240ドットの6インチ相当の液晶画面を装備したもの。通信速度は「MediaX 1200」が 1200 bps，「Media

MediaX2400

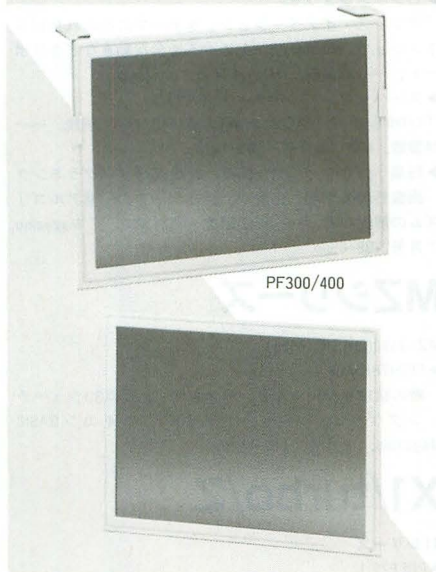
X2400」が2400bpsである。

この通信機能を使い，パソコン通信やビデオテックス，FAX通信（MediaX2400のみ）など，さまざまな双方向通信に対応できる。

価格は「MediaX1200」が29,800円（税別），「MediaX2400」が36,800円（税別）となっている。

〈問い合わせ先〉

マイクロコア(株)　☎03(3448)0811

覗き見防止機能つきCRTフィルタ
PF300/400
住友スリーエム

PF300/400

住友スリーエムは，覗き見防止機能つきCRTフィルタ「PF300/400」を発売した。

覗き見防止機能は，極微細のルーバーを組み込んだライトコントロールフィルタをはさみ込むことで実現している。このフィルムによりCRT画面の視角を約60度に制限し，左右からの覗き込みをしても画面が真っ黒になりCRTに表示されている情報を読み取れない。

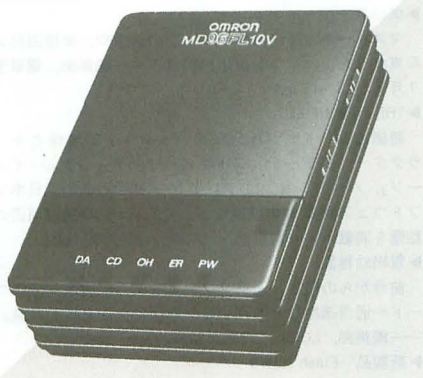
また，CRTフィルタに施された特殊コーディングにより反射光を防止するとともに，CRTからの電磁波を99.9%カット。静電気も防止でき，ホコリの付着も防げる。

価格は，「PF300」が35,000円（税別），「PF400」が37,000円(税別)。ともに10～13インチ，14～15インチ用の2種類が用意されている。16～19インチ用も近日発売予定である。

〈問い合わせ先〉

住友スリーエム(株)　☎03(3709)8111

9600bps対応のインテリジェントモデム
MD96FL10V/XL10V
オムロン

MD96FL10V

オムロンは，移動体通信に対応した9600bpsのインテリジェントモデム「MD96FL10V」「MD96XL10V」を発売した。

両機とも従来機同様（MD24FLV/M/XL10V/TV通信速度2400bpsなど），エラー訂正機能にCCITT V.42とMNPクラス4を標準でサポート。そして，データ圧縮機能にCCITT V.42bisとMNPクラス5を装備している。これにより，通信速度の実効通信速度が最高で約3倍（28800bps）まで向上することになる。さらに，「MD96XL10V」では，EIAクラス1に準拠し，G3FAXとの送受信が可能なFAX通信機能も兼ね備えている。

本体サイズは，68mm（幅）×26mm（高さ）×93mm（奥行）とコンパクトで，据え置き，携帯でも使用可能。また，電源にACアダプタ，乾電池（単三乾電池使用）を使用することができ，電池駆動の場合は，連続2.5時間の連続駆動ができる。

価格は，「MD96FL10V」が42,800円（税別），「MD96XL10V」が46,800円（税別）である。

〈問い合わせ先〉

オムロン(株)　☎03(5488)3221，06(282)2672

X68030用増設RAMボード
SH-5BE4-8M
アイ・オー・データ機器

アイ・オー・データ機器は，X68030用増設RAMボード「SH-5BE4-8M」を発売した。

これにより，標準実装の4Mバイトから一気に12Mバイトまで増設することができ，グラフィック，アニメーションなどメモリを大量に消費する環境に対応できる。

価格は55,000円（税別）。

〈問い合わせ先〉

(株)アイ・オー・データ機器　☎0726(60)3366

INFORMATION

テピア・サマーフェスティバル'93 クルマと遊ぼう!!
機械産業記念事業団

機械産業記念事業団(TEPIA)は，TEPIA第6回展示「ADVANCED VEHICLE～21世紀のクルマ学～」(12月17日まで開催)の一貫として，夏期特別イベント「テピア・サマーフェスティバル'93クルマと遊ぼう!!」を開催する。

本展示では，車の先端技術や未来像だけでなく，車のもつさまざまな側面を遊びながら体験，学んでいくことを目的としている。ゲームコーナーやおもしろ自動車の展示，車関連特販コーナーなど，3階のイベントホールをメインにし，4階ではミニチュアソーラーカーの工作教室や親子安全教室の講演会などが行われる。

また，1階屋外の庭園では，RVを中心としたアウトドア教室など，駐車場では電気自動車の試乗会やソーラーカーのラジコンレースなども予定されている。

開催期間，開催会場は以下のとおり。

開催期間：7月31日（土）～8月8日（日）の8日間（月曜日休館）

開催時間：平日10：00～18：00
　　　　　土，日曜日10：00～17：00

開催会場：TEPIA（産業記念会館，営団地下鉄銀座線外苑前下車）

なお，入場は無料となっている。

〈問い合わせ先〉

TEPIA第6回展示事務局　☎03(3226)8356

FILES Oh!X

このインデックスは，タイトル，注記──著者名，誌名，月号，ページで構成されています。夏です。太陽と仲よくする季節です。海へ行こうか，それとも山へ？　思いっきり身体を伸ばしたあとには，パソコンで頭のトレーニングもね。

参考文献
I/O　工学社
ASCII　アスキー
コンプティーク　角川書店
C Magazine　ソフトバンク
テクノポリス　徳間書店
電撃王　主婦の友社
POPCOM　小学館
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶THE NEWS FILE
シャープの世界最小，最軽量のMDプレーヤー「MD-S10」と「MD-D10」。メガドライブ・メガCDの低価格化やパイオニアのLD-ROM「レーザーアクティブ」のソフト情報，JRAが提供する競馬のデータベースなど，パソコン関連の話題。──編集部，LOGIN，11号，30-37pp.

▶特集　夏だ！　飛び出せ！
世は3D。ゲームやCGも疑似的に立体に見せる技術や，本当に立体に見える工夫をいろいろと行っている。なつかしの赤青メガネからホログラム，偏光を使ったものまでありとあらゆる3D技術を紹介。音の3D効果もあるぞ。──編集部，LOGIN，11号，211-225pp.

▶電網幼稚園
パソコン通信の初心者ちゃん集まれー。実践的ネット入門講座。パソコン通信に必要な機材，普通の人にはちとわからないパソコン通信用語などを解説。──編集部，LOGIN，11号，226-233pp.

▶TV GAME CLUB
東京・二子玉川のナムコ・ワンダーエッグの新アトラクション「バーチャルビーグル」の紹介と攻略。──編集部，LOGIN，11号，254-255pp.

▶煩悩マーケット
マスターネットのパワーアップの話題や，集積回路の写真集など，パソコン周辺の記事。──編集部，電撃王，7月号，132-133pp.

▶THE NEWS FILE
超緻密なハイビジョンCGのパイロット版映像とキャラクター。CGアーティストグループ「ディジタル・イメージ」，ハイビジョンLDプレイヤー「HLD-1000」，日本ソフトウェア大賞'92の開催など，パソコンやAV機器周辺の話題を満載。──編集部，LOGIN，12号，32-39pp.

▶電網幼稚園
前号からの続き。通信ソフトとモデムの設定，漢字コードや通信速度などのパソコン通信に欠かせない知識。──編集部，LOGIN，12号，220-223pp.

▶新製品　Flash NEWS
シャープのフルカラーイメージスキャナ「JX-325M/235/235X，JX-23F12」など，周辺機器の新製品。──編集部，マイコンBASIC Magazine，7月号，70-74pp.

▶Bug太郎のプログラム・タイム　その7
「天秤ゲームにチャレンジ！」ということで，うまくバランスをとる重量シミュレーションゲームの作成で，BASICやプログラミングアルゴリズムを考える。──谷裕紀彦，マイコンBASIC Magazine，7月号，86-87pp.

▶BASICプログラミング講座　第15回
光の反射，屈折をプログラミング。物理現象のシミュレートをアルゴリズム化し，プログラムを解説。──東幸太，マイコンBASIC Magazine，7月号，88-92pp.

▶ASCII EXPRESS
5月に東京・晴海で開催されたビジネスショウ'93や，シャープのフルカラーイメージスキャナ「JX-325X」などの話題。──編集部，ASCII，7月号，206，222pp.

▶プリンタ用紙の研究レポート
インクジェットプリンタ「BJ-15v」を使い，いちばん美しく「にじませる」紙質のプリンタ用紙を探す。──国友正彦，ASCII，7月号，269-276pp.

▶特集　超小型パソコンの現状と未来　前編
「Subnoteがほしい！」。ノートパソコンよりさらに小さい「真の持ち運びできるマシン」を国内外から集め，日本語環境のポテンシャルを探る。──編集部，ASCII，7月号，285-300pp.

▶PRODUCTS SHOWCASE
高速になった新型バブルジェットプリンタ「BJ-220JS」，小型ページプリンタ「LP-1500」などの周辺機器を紹介。──編集部，ASCII，7月号，301-316pp.

▶中学生は知っている！
今年の4月から始まった，中学校の技術・家庭科においての情報処理教育「情報基礎」の授業内容について。──寸村剛，ASCII，7月号，317-323pp.

▶日本の機械式計算機のルーツを求めて……
計算具の歴史を研究している東京理科大学諏訪短期大学教授の内山昭氏にインタビュー。──遠藤諭，ASCII，7月号，328-331pp.

▶DIGI-VIS TODAY
デジタル音声/映像がもてはやされている昨今だが，その実態は明確でない。デジタルとアナログの位置関係がどうなっているかなどをソニーPCLにインタビュー。──聖咲奇，ASCII，7月号，390-391pp.

▶バカパパのモノを買い物
海外旅行のミヤゲモノ特集。中文キートップなどのあやしいグッズが勢ぞろい。──バカパパ，ASCII，7月号，392-393pp.

▶VISIMO
VISIMOとは，カメラで捉えた画像をRS-232Cを介してコンピュータに送る新しいタイプの情報ツール。そのパフォーマンスとポテンシャルを探る。──吉岡哲也，My Computer Magazine，7月号，102-105pp.

▶BUSINESS SHOW'93 TOKYO
5月19日から22日まで，晴海で行われたビジネスショウ'93 TOKYOの模様をメーカー別にレポート。──有坂静香，My Computer Magazine，7月号，128-131pp.

▶MYCOM WATCHING
高知医科大学で構築された「周産期医療支援システム」は，妊婦と胎児を守るための情報ネットワーク。その活用の模様をレポートする。──菊地秀一，My Computer Magazine，7月号，134-137pp.

▶パソコン買い方心理学　激得！ 秋葉原攻略法7
秋葉原で安く売られている商品のカラクリと見分け方をレクチャーする。──島川言成，My Computer Magazine，7月号，188-191pp.

▶コンピュータ博物館
今月はシャープの往年の名機「MZ-80K」を取り上げる。マニアックな設計思想が特徴だった。──Y.I，My Computer Magazine，7月号，202p.

▶ビジネスマンのための情報管理術　第11回
シャープ電子手帳シリーズの最高峰「PV-F1」の機能を詳細にレポート。──塚田洋一，My Computer Magazine，7月号，224-227pp.

▶ビジネスショウ'93
「オフィス・イノベーション」をテーマに行われたビジネスショウ。Windows 3.1をめぐる活発な動きなどをリポート。──編集部，I/O，7月号，84-85pp.

▶スーパーコンピューティング入門31
「1/fゆらぎ」と「フラクタル」の関係について解説。──林智雄，I/O，7月号，145-147pp.

▶特集　グラフィックを操る　画像処理プログラミング
画像処理の原理と応用や，画像データの圧縮アルゴリズムの解説など。──堀江郁弥・高村誠之，C Magazine，7月号，38-69pp.

MZシリーズ

MZ-2500(BASIC-M25)

▶FIGHTER ACE
敵の猛攻をかいくぐり，ボスを倒す，疑似3Dシューティングゲーム。──もったんSOFT，マイコンBASIC Magazine，7月号，110-112pp.

X1/turbo/Z

X1シリーズ

▶FREEZE！
倉庫番とフラッピーのまぜあわせ(？)アクションパズルゲーム。──森敬雄，マイコンBASIC Magazine，7月号，132-133pp.

X1turboシリーズ

▶X1turbo用ディスプレイ・テレビをアナログRGBに
デジタルRGBオンリーの「CZ-850DE」をPC-9801用アナログRGB対応に改造する。──新田隆幸，I/O，7月号，148-151pp.

X68000

▶NEW SOFT
未来の香港が舞台の超伝奇RPG「幻影都市」，戦闘シー

ンがハデな「銀河英雄伝説Ⅲ」。５月中旬以降に発売予定の機種別ゲームソフト一覧表も。――編集部，LOGIN，11号，14-15，29pp.

▶X68030新聞

１年半ぶりのコナミの新作「悪魔城ドラキュラ」，ビデオゲームアンソロジーシリーズ第４弾「リブルラブル」，スペースウォーシミュレーション「銀河英雄伝説Ⅲ」。――編集部，LOGIN，11号，242-243pp.

▶パソコンゲーム羅針盤

ゲームソフト売上ランキングや，機種別新作ソフト一覧表など。――編集部，電撃王，７月号，26-30pp.

▶電撃パソコン

お花畑で妖精をバシシするアクションゲーム「リブルラブル」。――編集部，電撃王，７月号，78p.

▶榊涼介のSLG巷談

シミュレーションゲームに関する連載。「信長の野望・覇王伝」など。――編集部，電撃王，７月号，82-85pp.

▶Software Hot Press

新作ゲームの紹介。話題沸騰アクション「悪魔城ドラキュラ」，なつかしの「リブルラブル」など。――編集部，POPCOM，７月号，20p.

▶SOFT EXPRESS

「リブルラブル」「ヴェルスナーグ戦乱」など新作ゲームを紹介。機種別ニューソフトインデックスも。――編集部，コンプティーク，７月号，69-74pp.

▶HOW TO WIN

「大航海時代Ⅱ」は先月の概要紹介に続き，実際のゲームの進め方を解説。「信長の野望・覇王伝」のリプレイはゲスト軍師にキューティー鈴木。――編集部，コンプティーク，７月号，98-101pp.

▶GAMING WORLD

７月に発売予定のアクションゲーム「悪魔城ドラキュラ」は，X68000版オリジナルのシーンを中心に豊富な画面写真で紹介。ほのぼのバシシのゲーム「リブルラブル」，ファンタジーRPG「ヴェルスナーグ戦乱」。――編集部，テクノポリス，７月号，26-27，34-35pp.

▶新作ソフト発売予定表

各機種の新作ゲームソフトを発売予定日，ジャンル別に分類。――編集部，テクノポリス，７月号，46-50pp.

▶NEW SOFT

６月上旬以降に発売予定の機種別ゲームソフト一覧表。「悪魔城ドラキュラ」は豊富なカラー画面で紹介。――編集部，LOGIN，12号，14-15，18-19pp.

▶最新ゲーム徹底解剖!!

「大航海時代Ⅱ」の人材紹介と実践プレー。――編集部，LOGIN，12号，116-119pp.

▶X68030新聞

「リブルラブル」「ヴェルスナーグ戦乱」ほか，「ズームスプライトエディター(仮)」「マージャンクエスト」など新作情報。――編集部，LOGIN，12号，200-201pp.

▶強い子ってい～いなっ!!　By神様

寒さに負けない強い子をつくるために子ども部屋の温度を調節するシミュレーションゲーム。――濱口和彦，マイコンBASIC Magazine，７月号，134-136pp.

▶マウス・レース

マウスの左右ボタンで操作するレースゲーム。――まんぞう，マイコンBASIC Magazine，７月号，137-139pp.

▶ATTACK TRAINING

ブルーセーバーを操作して，時間内にターゲット３つを破壊！　シューティングアクションゲーム。――林純一，マイコンBASIC Magazine，７月号，140-142pp.

▶STAR FOX　～ENDING～

任天堂のスーパーファミコン用ゲームのミュージックプログラム。要NAGDRV＋GS音源。――石原英治，マイコンBASIC Magazine，７月号，152-153pp.

▶SUPER SOFT HOT INFORMATION

格闘アクション「餓狼伝説」，「リブルラブル」「銀河英雄伝説Ⅲ」など新作情報。――編集部，マイコンBASIC Magazine，７月号，別冊付録10-11pp.

▶AV STRASSE

ウィンドウ環境を快適にするツールを収めたSX-WINDOWデスクアクセサリ集を紹介。――編集部，ASCII，７月号，358-359pp.

▶FREE SOFTWARE INDEX

主要ネットにアップロードされたソフトウェアから，秀でたものをピックアップ。大回転ゲームSKI.Xなど。――編集部，ASCII，７月号，427-433pp.

▶なんでもQ&A

SX-WINDOWに付属のシャーペン.Xでフォントの種類を知るにはどうするか，背景をグレーにするには，などの質問に回答。――シャープAVCシステム事業推進室，My Computer Magazine，７月号，242-243pp.

▶HOBBY EXPRESS

X68000芸術祭出展作品から５本のゲームをまとめた「ザ・ワールド・オブ・X68000」。――あゆさわかつみ，My Computer Magazine，７月号，256p.

▶GCCで学ぶX68ゲームプログラミング　第20回

ゲームの仕上げその３として，グラフィック画面のスクロールを使った背景の移動ルーチンを作成する。――吉野智興，C Magazine，７月号，129-134pp.

ポケコン

PC-E500

▶神経衰弱

コンピュータと記憶力の勝負。難易度の調整ができる１人用神経衰弱ゲーム。――MiPS，マイコンBASIC Magazine，７月号，144p.

新刊書案内

テクノバブル
ジョン・A・バリー著
岩谷　宏訳
工学社刊
☎03(3375)5784
A5判　491ページ
2,500円（税込）

わけのわからんテクニカルタームが氾濫し，用語の意味はどんどん曖昧になっていき，記述もまちまちで，こりゃ大変，どうしようか，なんて思ったとき，アメリカではどういっているのか，どう表記しているのかって調べてそれを参考にしたものだった。それでも困るときがある。特にパソコン界独特のニュアンスで使いがちな，基本的な語を説明するときだ。が，なんのことはない，アメリカでも同じように乱れていたのである。「テクノバブル」というのは，コンピュータ以外の分野でも使われるようになったコンピュータ用語をさし，「最悪の場合には単なる“ノイズ”であり，何も意味しないか，あるいはものごとを曖昧化するために」使われる言葉なのだ。本書は「コンピュータ言葉の発生起源を調べてそれの社会と言語への影響を考察すること」を目的としているという。なんとまあ，すごい本である。

技術者の間で使われているだけであったコンピュータ用語をテクノバブルとして一般にばらまくのは，主にマーケティングやPR部門だという。本書はそういったテクノバブルを槍玉に上げ，コンピュータ業界独特の悪文を示し，ひとつひとつの言葉を暴いていく。

問題は，本書が英語という言語を対象としていることだ。だから，最低限の英文法は知識としてないと面白みに欠けるし，日本人はこだわらないような点にまでこだわっている部分もある。これだけ，英語という言語に依存した内容だと，訳者も大変だったと思う。訳注も豊富に入っているが，英語のコンピュータ用語を皮肉ったジョークをそれが伝わるように訳するのは並み大抵のことではない。それでも，本書の訳出は十分に意味があると思う。

いやはや，アメリカでこれだから，さらにカタカナ化，日本語化が加わる日本で混乱を極めているのも当たり前なのだ。　(K)

やさしい
コンピュータ科学
アラン・ビアマン著
和田英一訳
アスキー刊
☎03(3797)3225
B5変形判
495ページ
4,800円（税込）

電脳映像世界の探検
田村秀行・
北村素子共著
オーム社刊
☎03(3233)0641
A4判　388ページ
2,900円（税込）

原題を直訳すると「コンピュータ科学における重要概念」。コンピュータ科学の講義のための教科書としてまとめられたものである。「概念」の解説書であるから，多くの入門書のようにビットやコンピュータの仕組みの話からアプローチを始めたりはしない。まずは，質問の連続から結果を導く「決定木」のアルゴリズムを解説するのである。コンピュータの「考え方」を学んだあとで，それを実現する回路，さらにはトランジスタの製造工程やマシンアーキテクチャへと，話を広げ，かつ掘り下げていく。簡単ではないが決して難しくはなく，無理なく学べる「教科書」といえる。

本書の中心となっているのは，光エレクトロニクスと画像工学の技術専門誌「O plus E」誌の連載記事。コンピュータによる画像処理技術の現在を知るために，イメージ処理の研究者10人に対し，それぞれの研究分野での根底となる考え方や取り組みの現状をインタビューしたものである。そこでは，医療現場，考古学研究，芸術，出版などそれぞれの現場での，論文には出てこないような研究者のスタンスや技術の実際が語られている。

単行本化にあたり，専門外の人々の理解のために，モーフィングやバーチャルリアリティ，画像データ処理の意味などの解説が加えられている。

QUESTION and ANSWER

あやしいRED ZONEのことですが（別にRED ZONEに限らなくてもいいが），Macintoshの2HDフォーマットが読めるんでしょうか？

京都府　松田　英弘

「Macintoshの2HDフォーマット」というのがなにを指しているのかわかりませんが，Human68kではMacintosh系フォーマットをアクセスすることはできません。MS-DOSフォーマットの場合，X68000で使っているような1.2Mバイトはサポートされていないようです。そして1.4MバイトフォーマットはX68000CompactXVIおよびX68030Compactの3.5インチFDDではアクセスすることはできません。

Human68k ver.3.0に付属のFDDEVICE.Xを使用することでソフトウェア上では1.4Mバイトフォーマットを使用できるようになるのですが，肝心の内蔵ドライブではこれには対応できません(FORMAT.Xも/4で対応はしている)。なぜ使えもしない機能がサポートされているかというと，当初はHuman68k ver.3.0で1.4Mバイトへの対応を行う予定だったようですが，ドライブの改造と再調整の手間がかかりすぎるので採用は見送られたためのようです。ちょっとしたドライブ改造で対応できるようになるということですが，かなり微妙な調整が必要になるらしく，素人は手を出さないほうが賢明でしょう。

ちなみに，この1.4Mバイトフォーマットというのは3.5インチドライブでだけ存在するMS-DOSフォーマットです。混同している人もいるようですが，一般に2HCフォーマットと呼ばれているものとは別のものですので注意してください。さらに，最近はX68000の内蔵ドライブで1.4Mバイトフォーマットを扱うフリーソフトウェアも発表されていますが，IBMなどのものとの互換性はないようです。

ついでに，2DDフォーマットについても解説しておきましょう。X68030Compactのドライブでは2DDフォーマットの読み書きができますが，X68000CompactXVIではできません。また，ここでいう2DDとはMS-DOSの2DDフォーマット（720Kバイト）のことであって，通常のMacintoshフォーマット（800Kバイト）は読み書きできません。

これはMacintoshは回転数を可変にした特殊なディスクドライブで容量を稼いでいるためで，通常のディスクフォーマットとはまったく違ったものなのです。

AMIGAの世界では普通の固定回転数のドライブの電源をON/OFFして回転速度を変えるようなボードでMacintoshのフォーマットを読み書きするというものもありますが……。

とにかく，3.5インチ2DD（720Kバイト）フォーマットは世界中のあらゆるマシンで共通に使用できるディスクフォーマットです。使えないのはX68000CompactXVIくらいのものです。このモードさえ使えればファイル互換の問題の大半が解決されますので，2HDは気にする必要はないでしょう（実際，ほとんど使われていません）。

某○才ブックスの現在30号まで出ている本にDIS.Xについて記されていました。マシン語はおろかX-BASICでさえもマトモに知らないくせにディスクの中をDUMP.Xを使って未知との遭遇をし「生きてるって素晴らしい」などと（中略）Oh!Xの付録ディスクに入っていたと解凍しようとしたところ（中略）次にブチ当たったのがFEFUNC.Hが入ってないこと。(中略)環境変数includeをセットしろとのこと。なんだ環境変数って？マニュアルを見ると環境ファイルはあっても環境変数はない。おそらくCコンパイラのほうには書いてあるのだろうが（中略）実際，僕がわからないのは「環境変数」「include」ってなんですかということです。

神奈川県　平井　秀司

DIS.Xはプログラム解析用のツールですが，生成したソースコードをアセンブラに通すことを前提としているため，Cコンパイラなどのツールを持っていない人がいきなり使うのは確かに難しいかもしれません。

「Cコンパイラを買ってください」というのがいちばん手っ取り早いのですが，それではあんまりなので，解説します。

「環境変数」

Human68kのマニュアルでは「環境文字列」という名前で記載されています。しかし，Cコンパイラのマニュアルでは「環境変数」という呼び名で解説されています。どちらかといえば「環境変数」というほうが通りがいいので，以後は環境変数という呼び名で統一します。

で，この環境変数というものはなにかというと，

COMMAND.Xの機能のひとつで，

A>SET AAA＝環境変数です

のようにするとAAAという変数に「環境変数です」という文字列が格納されます。ただし，読み出すときには%AAA%のように%2個で変数名を囲んで参照します。

A>ECHO %AAA%

のように使うわけですね。

こうしておいて，コマンドラインやバッチファイルなどで%AAA%という文字列を使うと，自動的に「環境変数です」という文字列に置き換えられます。

これだけでは，具体的にどのようなときに使われ，どんな効用があるのかということがわかりづらいと思います。

質問にある状況で見てみましょう。C言語でコンパイルをするときには，コンパイラとソースプログラムだけでなく多くのファイルを参照します。

それらのファイルがある場所というのは使っている人によって違うことが考えられます。ある人はA:¥INCLUDEというディレクトリ内に置いているかもしれませんし，またある人はD:¥TOOLS¥CC¥INCLUDEという場所に置いているかもしれません。

SX-WINDOWなどでは場所がわからないときにはすべてのドライブの中をサーチして探してきます。しかし，開発中のプログラムというのはえてして同じファイル名を使ったりしますので，探し方によっては別のファイルと間違えることがあるかもしれません。Cコンパイラの起動時にコマンドラインで指定するという方法もありますが，いちいち指定するのは面倒です。

ここで活用されるのが環境変数です。適当な環境変数にファイルのある場所を入れておき，ツールから参照します。そういう「お約束」で開発ツール群は作られているわけです。

AUTOEXEC.BATでPATHを指定すると，指定したディレクトリにあるコマンドは場所を指定しなくても使えますね。あれと同じことです。実は，「PATH」コマンドというのは環境変数「path（小文字）」に文字列を設定する機能を持ったコマンドだったのです。COMMAND.Xは「指定されたコマンドがカレントディレクトリにみつから

なかった場合にはpathという環境変数内に書かれたディレクトリ内を順に探していく」というふうに作られていたわけです。

DIS.Xで参照しているファイルはCコンパイラが参照するファイルを流用しているので，Cコンパイラと同じようにファイルを検索するようにしているわけですね。Cコンパイラを使う場合には，必ず「include」「lib」という環境変数がセットされているはずですので。

で，対策ですが，DOSCALL.MAC, IOCSCALL.MAC, FEFUNC.Hの入っているディレクトリ名を“include”に設定するようAUTOEXEC.BATに，

A>SET include=～

のような行を加えてください。

DIS.Xをそんなに多用しない場合は環境変数を使う必要もないでしょう。「カレントディレクトリになかった場合」に，この環境変数に設定されているディレクトリを探しにいきます。ですから，DOSCALL.MAC, IOCSCALL.MAC, FEFUNC.Hをすべてカレントディレクトリに置いておけばちゃんと動作するはずです。

いくつかの環境変数はシステムで使用することになっています。これらの環境変数と同じものをほかの目的で使うとシステムが正常に動作しなくなることがありますので注意してください。COMMAND.Xで使っているものには，

path
temp

があります。pathはパスを設定するもの，tempはテンポラリパスの指定です。つまり，さまざまなツールが作成する中間ファイルを生成する作業場所の指定です。できるだけ高速なドライブを設定しておくとよいでしょう。RAMディスクにするのが一般的です。ただし，ある程度の容量がないと大きなプログラムをコンパイルしたりするときにエラーが出てしまいますので注意してください。

Cコンパイラで使っているのは，前述の，

include
lib

などです。ちなみにZ-MUSICでも，

zmusic

というのを使っています。

これらの環境変数は大文字と小文字を区別するので注意してください。

X68000PROを使っています。電源プラグを抜いているときにSRAMに供給されるバッテリーの電力はいつなくなるんでしょうか？SRAMに組み込んだプログラムが永久ループして困っています。修理に出すしかないのでしょうか。　北海道　成瀬　直人

SRAMを消去するためにバッテリーの切れるのを待つというのはおすすめできません。X68000での正確な期間はわかりませんが，X1のときはバッテリーが切れるまで半年でしたから。以前，電極をショートさせて放電したという無茶な人もいましたが，危険なので絶対にやめてください。

さらに実際には，バッテリー（蓄電池）が使用されているのはX68000の初期のモデルのみで，大半のものには一次電池（要するに普通の電池）が組み込まれています。最近のモデルでは簡単に取り外しできるようになっていますが，基板に部品として組み込まれているものも多いようです（ちなみにPROは基板に組み込まれているタイプ）。これらの電池の寿命は年単位です。まだ電池が切れたという話は聞いたことがありませんから，5年以上はもつはずです。

ということでハードウェアの面から考えるのはやめておきましょう。

ソフトウェアでSRAMをクリアするためのプログラムがリスト1です。SRAMで使用しているうちのプログラム領域だけを初期化します。アセンブラがない場合にはリスト2をMAC.Xで打ち込んで117バイトでセーブしてください。

SRAMにはシステムの起動設定などの重要な情報が格納されていますので，プログラムの暴走などでは簡単に書き換わらないようになっています。書き換える場合はシステムポートの5番に$31を書き込んでから作業を行う必要があります。

ちなみに，SUPER以降の機種ではCLRキーを押しながら起動するとSRAMの内容を消去してから立ち上がるように設計されています。それから一般的な傾向としてX68000PROというのはSRAMがどうも不安定ですので，SRAMにプログラムを格納しておくことはあまりおすすめできません。あまりにも頻繁にSRAMの内容が破壊されるようであれば修理に出すしかないでしょう。　（中野　修一）

リスト1　SRAMCL.S

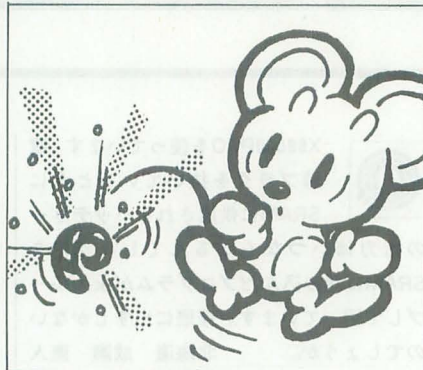
```
 1: ************************************
 2: *
 3: * ＳＲＡＭ領域をクリアする
 4: *
 5: ************************************
 6:
 7:   include a:¥c¥include¥iocscall.mac
 8:   include a:¥c¥include¥doscall.mac
 9:
10:   .even
11:   .text
12:
13:   clr.l    a1
14:  .IOCS     _B_SUPER
15:   * スーパーバイザモードへ
16:
17:   move.b  #$31,$e8e00d
18: * システムポート5に$31を設定すると、
19: * ＳＲＡＭ領域に書き込みが許される
20:
21:
22: * ＳＲＡＭのプログラム領域をクリア
23:
24:   movea.l #$ed0100,a0
25:   move.l  #$fbff,d1
26:
27: top_loop:
28:   clr.l    (a0)+
29:   dbra     d1,top_loop
30:   move.b  #$0,$e8e00d
31:   move.l  d0,a1
32:   IOCS     _B_SUPER
33:   DOS      _EXIT
```

リスト2　SRAMCL.X

```
000000  48 55 00 00 00 00 00 00 : 9D
000008  00 00 00 00 00 00 00 32 : 32
000010  00 00 00 00 00 00 00 00 : 00
000018  00 00 00 00 00 00 00 00 : 00
000020  00 00 00 00 00 00 00 00 : 00
000028  00 00 00 00 00 00 00 00 : 00
000030  00 00 00 00 00 00 00 00 : 00
000038  00 00 00 00 00 00 00 00 : 00
000040  93 C9 70 81 4E 4F 13 FC : F9
000048  00 31 00 E8 E0 0D 13 FC : 15
000050  00 00 00 ED 00 00 20 7C : 89
000058  00 ED 01 00 42 98 51 C9 : E2
000060  FF FC 13 FC 00 00 00 E8 : F2
000068  E0 0D 22 40 70 81 4E 4F : DD
000070  FF 00 00 00 00 00 00 00 : FF
000078  00 00 00 00 00 00 00 00 : 00
------------------------------------
CKSUM:  B9 45 A6 92 E0 75 E5 A6 B94D
```

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してくださいね。

宛先：〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク株式会社出版事業部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

さあ夏！　長い一日はどのようにすごしましょうか。早起きして散歩？　それとも夜更かしして，いちばんきれいな星を探しましょうか。帽子をかぶって，ちょっと遠くに行くのもいいね。山の太陽にも，海の大波にも負けないようにね。

◆リバーシってオセロと違うんですか？
横堀　正敏(29)埼玉県

さあ，英和辞典をひいてみてください。「othello」はなくて「reversi」はあるはずです。そう，オセロって登録商標なのですよ。ちなみに「Othello」はあるけど，これはシェイクスピアの登場人物ね。

◆MSX用S-OS“SWORD”が発表されることで，家にあるパソコンはすべてS-OS対応マシンになる(FM-7，MSX，X1turboII，X68000 PRO)。PASOPIAやSMC-777でも動作するプログラムを掲載し続けたのはすごい。徳永　賢一(25)愛媛県

◆本当に，ほんとうに，ホントーにMSX用S-OS“SWORD”を使えるのですね？　思い起こせば87年10月号P.131で落胆し，88年10月号P.145で希望を取り戻して以来，待って，まって，すっかり待ちくたびれて，いったいどのぐらい待ったと思っているのだ(計算してみましょう)。やっとこれで，あきらめずにしつこくOh!Xを購入し続けた「ご利益」があったというものです。良かったよかった。Turbo-Rだったらもっと速くなるのですか？　もしそうなら，私もタボちゃん買おうかな。MSX版がS-OSの活性剤になることを期待します。私も何か投稿したいと思っています。この次は，MSX用MAGICですね。期待しています。
P.S.　本当に出るんでしょうね。この際，少しぐらい遅れても文句はいいません。
林田　武之(30)長崎県

◆S-OSにそんなに深い思想があったとは思わなかった。わざわざ速度を犠牲にしてまで互換性を求める必要などあるのだろうかと批判的に見ていたときもあった。他人の真意をつかむのってやっぱり難しい。猛省。
高村　寿男(19)大阪府

MSXユーザーのみなさん，お待たせしました。ということで，7月号で紹介したS-OS“SWORD”ですが，ページの都合で残念ながら今月号にソースリストを掲載することはできなくなりました。配布方法については，P.105をご覧ください。今後の活用を期待して，デバッグ情報や投稿もお待ちしています。

◆オレンジ，青，赤とくれば次は緑か黄色でしょう(笑)。　宮本　淳(20)東京都

という，おはがきが何枚かきてるのですが……。えー，みんな何の話をしてるのかな，どーして緑か黄色なのぉ(笑)。べつに美奈子ちゃんとか亜美ちゃんとかファンの多い順に，というわけではない(と，思う)し(あれ，何の話だっけ？)。ところで，表紙の色にはチタンブラックとか金色とかリクエストはたくさんきてますが，金箔貼りのOh!Xじゃ受け取り拒否されちゃうかな。

◆6月号の特別企画「確率遊技シミュレーション」のプログラムのおかげで，ひさしぶりにキーボードにさわる気になった。
加藤　昌和(34)新潟県

◆半年くらい前の話，教室で黒板を点数表にしながらカード麻雀をしている生徒がいた。そこを通りかかった数学の某先生，
「何してるんだー」
「確率と統計の勉強です」
先生は苦笑して行ってしまった。

石井　大輔(18)東京都

そうそう，やっぱり実用に即してないと，頭に入らないものなのよね。何事も。

◆最近どうもスロット運が悪い。そこで6月号の特集を打ち込んで精進しようと思ったけれど，元祖ものぐさ太郎の私は10行も打たないうちにパチンコ屋に向かっていた。
そして結果は……。
おとなしく続きを打ち込んでりゃよかった。
新野　太郎(20)東京都

打ち込んでるうちにツイてくるかも。

◆横内氏は言葉がキツイ。いつか背後を襲われそうなところがある。しかし，それが氏の長所であり，これからもこの調子でさらなる活躍をしていただきたい。応援する。がんばれ！
坪田　雅己(17)広島県

◆母はパチスロ名人だ。行くたびに数万円は勝ってくる。最低でも元手は取り返す。本人いわく，勝負の決め手は「コツをつかむ」ことだそうだ。　河田　賢治(24)茨城県

◆確率遊技のシミュレーションですか。パチスロの乱数はよく知りませんが，パチンコの乱数は単純に＋1してるのが多いみたいですよ。割り込みごとに＋1して，入賞口に玉が入るとその値を取り出すので，乱数として十分使えるみたいです。　宮越　良幸(21)神奈川県

6月号の特別企画についてはぎゃんぶらぁな人からも，そうでない人からもいろいろおはがきいただきました。確率に関する問題は，パソコンで遊ぶにはなかなか面白い材料ですね。うまくプログラミングできた人はぜひぜひ投稿してね。

◆自動二輪学科試験に落ちた(埼玉県のは特に難しい)。家に帰ると僕のX68000の1ドライブが壊れてた。去年のいまごろも壊れたけど，自力で開けて直した。いまは直す気力もございません。……が，2日たって，テレビタイマーとして立ち上がった愛機はディスクをはき出した。自己修復機能付きパソコンX68000。
村上　洋樹(17)埼玉県

落ち込んでる持ち主に迷惑をかけまいとして，自力で直ったのね。けなげなX68000。二輪の免許取ったあとも，ほったらかしにしないでね。

◀岡村　直也　兵庫県
マシンをいじるときにはご注意ください。「香ばしいかほり」「カラカラ軽快な音」「ぽつんとりのこされた何かの部品」……みなさんも，をを。

◀小川　伸輔　宮城県
SUPERちゃんは封印しちゃうけど投稿は続けてくれるとのこと。受験関係でネタができるって……えらい！　それくらいたくましくなくちゃね。

しないでかわいがってあげてね。

◆最近，愛機の調教に成功しつつある。いままでは3日に1回しかプロテクトを外してくれなかった0ドライブが，いまでは電源ON一発目は100%書き込み可能になった。しかし，ディスクを入れ換えるともうアウトだったりする。

神谷　正樹(19)愛知県

X68000って……？

◆歳をとったせいだろうか。最近ゲームに興味を覚えなくなってきた。電脳倶楽部のゲームの解凍もせぬまま，ツールのみ解凍している。いまだにやっているのは音楽とCD(最近始めた)ばかり。何かを捨てねば何かは得られないのだろうか。　宮原　強志(30)鹿児島県

う～ん。1日24時間しかないのは変わんないから，あれもこれも，ってわけにはいきませんよね。そのときどきでいちばんやりたいことをするしかないのかも。

◆中学のときの同級生に会った。「人間ぽくなった」といわれた。どういう意味なのだろうか。

音羽　進(18)宮城県

そうか，Oh!Xには「人間に接近中」の読者もいたのか。ということは，ほかにも「人間寸前」「人間直後」「人間周辺」といろいろな読者がいるのかな。

◆私の母は現在，Jリーグと大相撲とみずいろぞうさんにハマッています。どうしたらいいでしょうか。　茂木　伸(23)神奈川県

やはりですね，家庭平和のためにも「一緒にハマる」のが息子としての正しいあり方というものでしょう。

◆7歳下の妹が毎日のようにマンガを描いているので，こっそり見てみると，これがバカげていて面白い！(本人は真剣ですが)。みんなに伝えられないのが悲しいです。ちなみに代表作は「ミナクル　ガールズ」「まさお君とプチちゃん　夏は食べほうだい」など。現在は「はつらつコメディ　スポーティガール」を連載中です。

中島　貴史(17)滋賀県

妹さんはいま10歳ですか。これは将来有望かも。もしも彼女が有名漫画家になったら，アナタは有名漫画家の兄！　すごい！(？)

◆X68000をパソコンだと思うのは大間違いである。これはシャープが我々をシャープのまわしものにするための……。あなたにも心当たりがあるだろう。X68000を買ってから，なぜかシャープという言葉に反応したり，知らないうちに電化製品がシャープ製になっていたり，吉田栄作がナイスガイに見えてきたりしたことがないだろうか。もしかすると私のようにシャープなしでは生きられなくなるかもしれない。

秋山　真一(19)茨城県

そういえば，ウチにもシャープ製品が着々と増加し続けているような……。こないだ買った掃除機にもあの見慣れた「SHARP」の文字が……あああ，そうだったのか。

◆最近，はやりの裸眼立体視が楽勝でできるようになったので，大学生協に置いてある本2冊分をぶっ通しで見ていたら，目が回り頭がクラクラしてきて死にかけた。頭をおさえつつ，フラフラと出ていく私は，周りから見ると二日酔いに見えたに違いない。まったくマヌケだった。

岡部　英隆(20)奈良県

◀芹澤　あや子　東京都
さてさて今月の「移植希望」はがきはコレ。「ミーハーな選択」なんていってるけど，いーじゃんいーじゃん。スキなものはスキなんだよねっ！

◀岩瀬　貴代美　福岡県
こらこらキミたち，Oh!Xを伝言板にしないように。まあ今回は，可愛いちきぺと笹美ちゃんに免じて(？)許してつかわそう。よしよし。

裸眼立体視って，やはり「時」と「場所」を選ぶかもしれないですね。あんまりのめりこむとアッチの世界に行っちゃうかも？

◆ああ……あと1カ月で彼女いない歴21年目突入か……(6/10現在彼女なし)。

杉山　洋之(20)埼玉県

ま，こういうのは「量より質」だからして，すてきなひとが現れるまで待つしかないですね。でも，もし好きなコがいるんなら，ぼーっとしてないでがんばるのだぞ！

◆クレーンゲームの人形がたまってしまって(450個ほど)，置き場所がない(笑)。

鈴木　広志(25)栃木県

450個！　累積投資額はいったいいくらになるんでしょう？　で，まさか全部同じ人形だったりしないよね……。450個のうさぎちゃんとか，450個のダルシムとかってちょっとぶっきー。

◆私はセーラームーンの下敷き欲しさに某アニメショップへ行きました。そこは，パソコンとはまったく次元を異にする，なんともアヤシイ空間だったのです。セーラーマーキュリーの等身大ポスターもできれば欲しかったのですが，私は20歳だし……。　斉藤　徹(20)東京都

某ライター氏の部屋には等身大「兄貴」が貼ってあるそうです(もちろん自作ね)。アニメショップとどちらがアヤシイかなぁ(五十歩百歩という説も)。でも，編集室はアヤシくないから大丈夫さっ！

◆後輩に高校生の彼女ができたそうな(彼は21歳)。「いやー，高校生に手を出すのは犯罪みたいでちょっとね」とかいってる。そうかな。私だったら中学生でもやっちゃうけど。まぁ，小学生だったら躊躇はする。中村　健(23)埼玉県

えぇえ，どーしてぇ？　べつに小学生と手つないだって，いいじゃない。え？　違うとこに手を出すのぉ？　僕わかんなぁい。

◆う～ん，世間は狭い！　最近わかったことですが，僕の友達の友達の友達が西川善司さんだそうです。友達に聞いた話によると，西川さんは毎晩，赤のプレリュードを乗り回しているとかいないとか。　小田　直樹(20)東京都

◆浦川氏が友人Aの友人で，金子氏が友人Bの友人であることがわかった。ポインタのポインタな気分。世の中狭い。　三森　浩一(24)東京都

「世界はひとつ，人類はみな兄弟」って？　うーん，それにしても悪いことはできないものですね(私はしないからいいけどさ)。

◆ゲームソフトのディスクのタイムスタンプを見ると，スタッフの生活がわかる……。

八尾　唯仁(16)神奈川県

スタッフのかなしい生活に涙して，感謝しつつゲームを楽しんでください。Oh!Xにはタイムスタンプは入ってないけど……。

◆ぼくの名前はセーブマンではないぞ。

西部　満(12)愛知県

え，正義の味方みたいでカッコイイじゃん。訳すと「救済者」ってのかな。

◆最近，女の子なX68000のイラストが多いですけど，「兄貴」なX68000のイラストが送られてきたら載せるのでしょうか？　え？　私のXVIは「さとみ」って名前のれっきとした女の子ですよ。　浜名　進(22)広島県

◆だからその，どこをど一間違ったら，あのドス黒くソソリ立ったSUPERが女の子に見えるのか!?　説明してもらおう。

上村　こーじ(24)広島県

ほら，人間でも何でも♂と♀がいるんだから，パソコンだって……。浜名さんちや小川さんちにいるのは女の子だそうですが。だから，「兄貴」なX68000のイラストだってピーなものじゃなければ，ちゃんと載せちゃうよん。

◆大学受験には見事に落ちました(失敗しました)。東京の某私大W大学も受けました。別に東京に住んでるからいいんですけどね。でも，僕の父親がいうには「浪人も無駄ではなく，お前の人生にとって必ず役立つ」ということだそうです。X68000やスーファミの理解もある父親にとても感謝しています。　尾形　敦(18)東京都

そっ，何事も経験ですよ。でも，それを活かすも殺すもそれはアナタ次第。がんばる

のだぞ。

◆6月号のSTUDIO Xの片平，磯田両氏によるミンキーモモの呪文についてですが，彼女はよく，呪文を省略して変身したものです。そんな，ややこしい言葉なしで使えるようなSX-WINDOWに期待しています。だから，じゃんじゃんアプリが出るといいなと思う今日この頃です。でも，ティラクルラミカル石(MPU)動け的楽しさもパソコンには必要ですね。　高濱　均(18)岡山県

うんうん，3歩進むとすぐ忘れるオツムの持ち主のワタクシといたしましては，呪文は簡単じゃなくちゃね。

◆6月号STUDIO Xで「24の人格」うんぬんと書かれていましたが，「24人のビリー・ミリガン」のことで間違いないと思います(「ワンダーゾーン」を観た弟から聞いた)。私の勤め先で印刷したので読んでみましたが，かなりスゴイです。でも「5番目のサリー」(でしたっけ？)を読んでからのほうがいいかも。まぁ，個人的には「アルジャーノンに花束を」のほうが好き。

須田　泰弘(23)埼玉県

キイスの小説ではたぶん「アルジャーノンに花束を」がいちばん有名ですよね。関係ないけど，某編集部には「アルジャーノン」と「次郎吉」というマウスがあって，命名者はX68000ユーザーだったりする……。

◆最近マンガ家のあいだにMacintoshがはやっているようですが，そのほとんどすべては見るに堪えないものです。手間はかかっているのでしょうが，あのノッペリしたグラデーションにはゲンナリします(よけい手抜きに見える)。ビッグTにしろそうです。僕が唯一認めるのは，都築和彦氏だけです。Macintoshじゃありませんが。　山川　剛信(21)福岡県

うーん，たしかにパソコンによるマンガのカラーリングは，いまの段階では単にテクニックを駆使してるだけで，その人の個性なんかはあまり見えてこないような気がします。まあ，技術点，芸術点ともに今後に期待ってとこでしょうか。

◆都築さんといえば，「ザナドゥ」や「ロマンシア」や「イースII」で有名ですね。しかし，最近の彼の描く女性キャラにアブナイものを感じてしまうのは僕だけでしょうか？

伴　武士(22)千葉県

そのアブナさにファンが急増中，なんてね。ところで，6月号に「都筑」さんって書いちゃったのですが，みなさまご指摘のとおり「都築和彦」さんです。チェックが甘かったなぁ。で，ごめんなさいなのです。

◆「なぶる」ってどういう字を書くか知ってますか？　「嬲」と書くのだそうです。いやぁ，古代の中国人って結構ブラックですね。中国史を勉強していると，いろいろ覚えますよ。え？　関係ないって？　こりゃまた，失礼いたしました。　矢元　章夫(19)兵庫県

「嬲」は「男が女につきまとう」というのを意味してるそうです。だから，両側の男は同一人物かもしれませんね。なんか過激なこと想像した人いるでしょ？　でもでもぢつはですね，「嫐」って字もあって，こちらも「なぶる」。ちなみにJISコードは554Bと554Cだよん。ね，パソコンでも表示できるでしょ？　だからどーしたって？　こりゃまた，失礼いたしました。

◆コプロを買った。しかし，Macintosh用なので宝石箱がついてこない。シャープさん，1,000円ぐらいで別売りしてください。

鈴木　武虎(19)愛知県

「買います」のコーナーに投稿してくれたら，誰か売ってくれる人がいる……かも？　ところで，宝石箱は何に使うのかな。

◆ようやくJリーグが始まりましたが，わが地元のガンバはどうも調子が出ない。誰かどうにかしてくれ～！　ガンバがんばれ！　ところで編集部でも盛り上がってますか？

松本　高佳(19)大阪府

◆がんばれ！　鹿島アントラーズ!!

信太　徹(23)神奈川県

信太さんのはがきを見たとき，茨城県の人かと思ったのですが。まあ，地元以外にもファンは当然いますよね。編集室でも，某チームのグッズとか持ってるサッカーファンの某氏が旗を持ってうろうろしてます。

◆こないだテレビ(広島ローカル「がんばれカープ」)を見ていたら，広島カープのブラウンが出ていた。司会者が「Jリーグに負けないよう野球を盛り上げてください」といっていたが，ブラウンはサンフレッチェの帽子をかぶっていた。

八谷　忠男(19)広島県

帽子を見て，「サンフレッチェを応援するひまがあったら練習でもせい！」といいたかったのか。ところでサンフレッチェの「サン」って「3」なんですよね。

◆編集って大変じゃないですか？

伊藤　治(17)岐阜県

うう，よくぞ聞いてくれました。それはそれは聞くもナミダ，語るも涙の物語。と，思わずよよと泣き崩れるあちきでありんす。なんていってたら信じてくれるかな……。まあ，忙しいなかにも一筋の喜びなるは，読者さまのおはがきじゃわいなぁ。ってこれは本当だよん。

◆最近，荒俣宏さんの「ワタシ no イエ」というホラー小説を読みました。そのなかでシリコンチップに生えるカビのエピソードがあって，これからの梅雨のシーズンを考えると恐怖を感じます。ウチのマンションは特に湿気がひどくソファの裏のカベには毎年カビが生えるし。恐るべしカビ！　橋本　和典(26)東京都

その恐怖が現実となった人が↓に……。

◆X68000が生体ウイルス(カビ)に襲われてしまった。電源入れてなくても感染していくから，この点に関してはコンピュータウイルスよりもたちが悪いな。　中村　健英(23)東京都

湿気はからだに毒なのね。シリカゲルなんか入れとくと効果的？

◆最近，とても眠くて眠くて会社ですご～く長い時間眠っているような気がする。でもデータはちゃんと入力しているし，はかどっているんだよなぁ～。そんなことってありませんか。浅い眠りだからかな？　弥山　宏一郎(24)熊本県

睡眠学習ならぬ睡眠労働ね。でも，ねぼけてへんなでーたが入ってたりしませんか？　おかしな感じ変換とか……ZZZ……。

◆近頃，寝ては職場の夢，起きては睡眠不足で，起きているのか眠っているのかわからない日が続いている。しかも，単調な生活が続くためか「以前もこんなことがあったような……」と思う日ばかり。何も考えないでいるときがいまいちばんの幸せかなぁ。　藤原　彰人(23)岡山県

夢のような毎日ですか。いいなあ。

◆みんなで「グリッドマン」を見よう！　6月5日の放送では，われらがOh!Xが映り，その1週間前にはX68000 PROが出演していたぞ！

小林　裕昭(23)東京都

えっ，Oh!Xが？　……ところで，グリッドマンって何ですか。ちっとも知らなかった担当者は見ていないのですよ。しくしく。詳しい情報，お待ちしてまぁす。

◆5月18日，朝のニュースを見ていたら，Windows ver.3.1のことを紹介していた。しかし，そのなかでテレビに唯一でかでかと映ったコンピュータがなんと，X68000だったのである(たぶんXVI)。WindowsとXVIって……？

◀佐田　匠　千葉県
「とにかく夏！」そうそう，夏ときたらやはり海。海ときたらやはり……。大量投稿してくれる佐田さんですが，今月は季節感あふれるこれに決まり！

◀武田　正道　兵庫県
武田さんちは海水浴場の近くで，水着のままで海まで行けるとのこと。じゃあじゃあ，近くにはこんな娘がいっぱいいるの？　う，うらやましい。

野村　雄一(17)奈良県

う～ん，それは「人目をしのぶ関係」。冗談はさておき，それはきっとパソコンを何も知らない人が「見栄えのするマシン」を選んだんですよ，ね。とすると，登場するのはやはりX68000！　なんてね。

◆なんだか知らないけれど，別に布教活動をしたわけじゃないのに，僕の周りでX68000を欲しがる人が増えてきている。しかもそのうち2人は，実際に購入してしまった。

「どうして？」

「だって，楽しそうなんだもん」

してみると，最強の布教活動とは，自分が楽しく使ってみせることなのかもしれない。

清水　智明(25)神奈川県

やはり，その人の心のなかから自然に興味が湧いてくるようになれば，本当の面白さや楽しさをわかってくれるような気がします。「北風と太陽と旅人」の話じゃないけど，人に何かを勧めるって，そういうことなのかもね。

◆何か，ロマンを求めるようなものが欲しい。

中村　吉邦(21)神奈川県

これこれ，そんなこといってぐずぐずしてないで，探しに行かなきゃだめじゃない！宝物は冒険の末に勝ち取るものなのさ。

◀近藤　隆生　埼玉県
ううん，ゲームではちまちましたサイズだけど，戦ってるアリスちゃんって，きっとこんな感じだよね。大胆不敵なホホエミがちょっとイカス！

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できないこともあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

売ります

★X68000用カラーイメージユニット「CZ-6VT1」を25,000円，RGBシステムチューナー「CZ-6TU」を15,000円，200Mバイトハードディスク「LHD-FM200E」を50,000円で売ります。箱，マニュアル，付属品はすべてあります。連絡ははがきでお願いします。〒370　群馬県高崎市江木町1646-2　明石マンション305号　岡田　修一(26)

★RGBシステムチューナー「CZ-6TU-BK」を15,000円，24ドット熱転写カラー漢字プリンタ「CZ-8PC3-BK」を15,000円で売ります。また，YAMAHA製MIDIキーボード「V2」とROM 2個とケースをセットで30,000円で売ります。詳しくは，電話番号を明記のうえ，往復はがきでご連絡ください。〒194　東京都町田市藤の台団地2-10-102　青木　恭一郎(19)

★アイ・オー・データ製拡張スロット用2Mバイト増設RAMボード「SH-6BE-2」を送料込み20,000円前後で売ります。希望価格を明記のうえ，往復はがきでご連絡ください。〒244　神奈川県横浜市戸塚区南舞岡2-12-3　野沢　幸弘(18)

★アイ・オー・データ製X68000拡張スロット用1Mバイト増設RAMボード「PIO-6BE1-A」と，X68000 PRO用内蔵20Mバイトハードディスクを，各10,000円で売ります。連絡は往復はがきでお願いします。〒501-61　岐阜県羽島郡柳津町佐波4641-1　奥田　弘一(18)

★Roland製MIDI音源モジュール「MT-32」＋システムサコム製MIDIボード「SX-68M」(ミキサー内蔵型)を40,000円以上で売ります。どちらも箱なし，マニュアルあり。買い取り希望価格を書いて，往復はがきで連絡してください。〒535　大阪府大阪市旭区今市1-5-15　大又　義嗣(19)

★21インチカラーディスプレイ「CU-21HD」(黒)＋RGBシステムチューナー「CZ-6TU-BK」(黒)をセットで60,000円以上で売ります。どちらも箱なし，マニュアルあり。バラ売りは不可。ディスプレイはスピーカー取り付け用部品が欠落しています。また，X1用ドライブユニット「CZ-502F」を10,000円程度で売ります。箱，マニュアル，インタフェイスあり。なるべく高く買ってくれる方，送料を負担してくれる方を希望します。連絡は往復はがきでお願いします。「CZ-502F」は価格相応のものとの交換でもかまいません。まずはご連絡ください。〒757　山口県厚狭郡山陽町埴生正寺830-5　藤本　格(22)

★X1，X68000用24ドット熱転写カラー漢字プリンタ「CZ-8PC3」を20,000円で売ります。箱以外の付属品はすべてあります。おまけで，取り替え用印字ヘッドを付けます。値引き可。ただし，高く買ってくれる人を優先します。連絡は往復はがきでお願いします。〒903-01　沖縄県中頭郡西原町千原59 琉大男子寮北辰C312　長田智和

★X1用FM音源ボード「CZ-8BS1」を12,000円前後で，またX1turbo用「NEW Z-BASIC」＋64Kバイトバンクメモリボード「CZ-141SF」を12,000円前後で売ります。どちらも箱，付属品はすべてあります。希望価格を書いて，往復はがきで連絡してください。〒182　東京都調布市染地1-1-39-6-202　竹原　久(22)

買います

★X68000 XVI用の2Mバイト増設RAMボード「CZ-6BE2A」を20,000円で，「CZ-6BE2B」がセットならば40,000円で買います。連絡は往復はがきでお願いします。〒950-21　新潟県新潟市大学南1-6675-2サンシャイン新大1F-1　黒田　雄一郎(19)

★X68000用拡張I/Oボックス「CZ-6EB1-BK」(黒)を40,000円以下で買います。希望価格を書いて，はがきでご連絡ください。〒157　東京都世田谷区北烏山4-31-10　桜井　暢(39)

★HAL研究所製のハンディスキャナ「Fine Scanner X68」を10,000円，または，オムロン製のハンディスキャナ「HS-10R2」を15,000円で買います。送料込みでお願いします。箱はなくてもかまいませんが，付属品，説明書などはすべて付けてください。連絡は往復はがきでお願いします。〒383　長野県中野市中央2-8-18　土屋　真一(17)

★ネオコンピュータシステムのサブCPUボード「POLYPHON」(8M)を70,000円前後，HAL研究所製のハンディスキャナ「HGS-68」を12,000円前後，SASIタイプのハードディスク(何Mバイトでも可)を適価，インテリジェントコントローラ「CZ-8NJ2」を10,000円前後，SCSIボードを15,000円前後でそれぞれ買います。大切に使いますのでよろしくお願いします。送料はこちらが負担します。連絡は往復はがきでお願いします。〒315　茨城県新治郡千代田町下稲吉1405　浜田　淳(19)

★X1用カラーイメージボードII「CZ-8BV2」を15,000円以内，RS-232C・マウスボード「CZ-8BM2」を10,000円以内，FM音源ボード「CZ-8BS1」を10,000円以内で譲ってください。すべて完動品で付属品，説明書付きのものを，送料込みでなるべく安価でお願いします。往復はがきでご連絡ください。〒168　東京都杉並区宮前2-18-13　溝渕　道也(24)

バックナンバー

★Oh!X1988年12月号を送料込み1,500円で買います。特集の部分がしっかりしていれば，多少の汚れはかまいません。連絡は往復はがきでお願いします。〒179　東京都練馬区光が丘6-1-4-1308　深澤　達(15)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今月は6月号の内容に関するレポートです。第8期のモニタの皆さん，1年間，ありがとうございました。

●6月号の特別企画は，なんだかソフトな内容でしたね。私としては「運だめし」的なゲームよりも理屈で攻めるゲームのほうが好きです。どちらかといえば，今回はゲームを作るという記事よりも，疑似乱数の生成の話が面白かったです。ひと口に乱数といってもいろいろあるものですね。あと，11周年特別企画としては，中途半端な気がしました。

安井　百合江(19)　X68000 PRO　愛知県

●7月号の特別企画ですが，ギャンブルということを前面に出しながら，乱数を使ったゲーム作り志向になっていてなかなか楽しかったです。個人的には運をコンピュータに任せるのが少々シャクですが，乱数というコンピュータならではの機能を使ったギャンブルもまた楽しいものといえるのではないでしょうか。全体としては，ちょっと中途半端であった感を拭えませんが，ギャンブルという人間の本質を刺激するものを取り上げられたことは，興味深いものだったと思います。

前田　秀樹(19)　X68000 XVI, Macintosh IIvx　京都府

●確かに特別企画の記事はどの作品も「確率遊技」には違いありません。しかし，どのように自分の作品が確率と関わっているのか，どのようにプレイヤーと確率が絡んでくるのか，ということにも触れてほしかったです。また，一様でない確率を用いた遊技もシミュレートしてほしかったな。

高橋　毅(22)　X68000 PRO, MSX2　埼玉県

●「こちらシステムX探偵事務所」を読んで，やはり，柴田氏を止められる人はこの世に，いやあの世にも存在しないのではないか，そう思いました。「いたってええやん。こんなライター」と思わせる柴田氏には，ぜひ生き残ってほしいものです。「マシン語カクテルinZ80'Bar」のときは，アルゴリズムの解説やプログラミングの思考などで，貴重なヒントを与えてくれたことがしばしばありました。これからこの連載でも学ぶことが多くなりそうで，いまからワクワクしています。

中矢　史朗(22)　X68030, X68000 ACE-HD　愛媛県

●7月号で新製品レポート「SC-55mkII」を読んで，いつのまにかSC-55が標準機のようになっていたとは思ってもみませんでした。結局，MIDIが信号レベルでの互換性しか保証されていないから，ローランドのような大手のマシンが標準機のように扱われてしまうのでしょう。まあ，音色番号と音色の統一がとれているのなら問題はないのかもしれません。願わくば，ころころと仕様を変更されないであってほしいです。記事内容については，必要な情報を押さえていてよかったです。

内藤　陽一(26)　X68000, PC-9801NS/E　東京都

●大人のためのX68000「第4回Oh!Xアンケート分析大会」を読んで，いまさらながら「これからのX68000は苦しいかもしれない」という勘ぐりをしています。まず，X68000ユーザーの割合ですが，一見すると所有率が上がってよかったね，と考えられます。しかし，現実にはそれほどX68000は売れていないと考えられるし，現在では過去ほどの魅力をもっていないという事実があります。このままの状態が続けば，結局，X68000の所有率は変化がないか，下がってしまうでしょう。そして，X68000 Compactについては，文章中でX68000 XVIに及ばない，ということが書かれています。まさにそのとおりで，これは他機種がグングンレベルアップしてX68000だけがおいてきぼりをくっている，ということを具体的な数字で示したものだと思います。また，Oh!Xも他機種ユーザーを取り込んでいくような魅力のある誌面にしていかないと，いくらX68030の所有率が上がってもほとんどが買い替えユーザーで発行部数が伸びない，となってしまうでしょう。自己満足ではなく，大いなる野望を抱いてもらいたいものです。

湯沢　聡(30)　X68030, X68000, X1 turboIII, MZ-2531/2861, PC-1360K, PC-6601, MSX/2　埼玉県

●私は，平行法も交差法もできません。主人はどちらもできます。以前，ステレオグラムを見せようとした主人に「ステレオグラムなんかやらないよ」といってしまいました。でも主人はめげずに「ほら，これなんかきれいだよ」とかいいます。なぜこんないい方をするのか，7月号の知能機械概論を読んでやっとわかりました。そっかあ，立体視ができると妙に嬉しいのか。だからあんなに流行するんですね。最近，娘が「お母さん見えないの？お父さんは見えるんだって」といいます。いいんだよ！　お母さんはふだんぼやっとしてるから，こういうのを見るときに目がぼやっとしないんだよ。あれ？

野原　志貴乃(31)　X68000 ACE-HD　埼玉県

ごめんなさいのコーナー

7月号　Oh!X LIVE in'93「赤い靴」

P.111　そのまま演奏すると楽譜どおりに演奏されず，テンポがおかしくなるバグが見つかりました。19行目の“t96”を“o96”と変更してください。

P.138　MSX用S-OS“SWORD”

表示行数変更ルーチンにバグがありました。7月号のままでは，MSX1かどうかチェックせずにVDPのレジスタ9を書き換えてしまいます。チェックするようにするために次のように変更してください。MSX1で0D4C$_H$番地を変更していた場合は，元に戻します。

```
0D4CH xx xx → 38 03
0D54H CC 10 → 54 29
2954H 47 3A 52 06 B7 78 C8 C3
      CC 10
```

そして，MSX1で25/24行表示を切り替えるための改造も掲載します。この改造を行うと“4”と“5”というコマンドが増設されます。“# 5”と入力すると25行もどきとなり，“# 4”と入力すれば元に戻ります。

```
2153H C3 1C → A8 1C
1CA8H FE 34 28 0B FE 35 28 07
      FE 36 28 03 C3 C0 1C D6
      1C 32 20 0D C3 3C 20
```

バグに関するお問い合わせは
☎03(5642)8182(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

いきなりCなんて リンダ 困っちゃ～う

▼困られてもどうしようもないのですが……ま，余談はさておき，今月の特集は「C言語実践的入門」です。とにかくリストを打ち込み体で覚える，知識なんてあとからついてくるもんさ。初めは形式だけを覚え，必要に応じてマニュアルをめくり知識の裏づけを行っていくようにすればいい。要するに，経験を積みながら知識を補っていこうね，という話です。ただし，この方法で問題となるのは，自分自身のやる気を持続させていかなくてはならない点です。やはり，なにかを作ろうという目的意識，みずから突き進む力がものをいいますからね。

ここまで熱くなれない人は，やはり地道にマニュアルを熟読し，参考書籍を読み知識を身につけましょう。その場合は，参考にした書籍，リストを自分の理解できる範囲で納得してください。そう，学び方なんて人それぞれ。自分にあった方法で学べばいいのです。

いままで興味のなかった人，始めてみようかなと思っている人も，プログラミング言語として，確固たる地位を築いたC言語の世界を一度はのぞいて見るのもいいと思いますよ。まず，やることを決め，１歩踏み出しましょう。あとは手探りでもなんでもかまいません。とにかく前へ向かって歩いていけばいいのですから。

▼いきなりですが，ここでお知らせです。お待たせしました。読者の皆さん待望の付録ディスクが10月号に決定！　内容については鋭意制作中，とだけいっておきましょう。まあ，あちらこちらでバスエラーやアドレスエラー，おかしな命令を出している現状ですからどうなることやら。

また，Z-MUSICについてもようやく発売のメドが立った，とようやく担当者が重い腰を上げたようです。今回，ディスク枚数の増加により定価が跳ね上がっていますが，それだけ充実したものをお届けできる。そう，確信しています。付録ディスク，Z-MUSICともに期待していてください。

投稿応募要領

- 原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。
- プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。
- 投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒103　東京都中央区日本橋浜町3-42-3
ソフトバンク出版部
Oh!X「㋢㋺㋯㋱」係

S H I F T · B R E A K

▶アクセラレータの犠牲になったかのように思われたX68000 EXPERTだったが，編集部に返しにきたとたんなぜか復活。再挑戦中。動作もしないうちからいうのも変な話だが，現在，ローカル32ビットバス化の際に使用するDRAMコントローラを物色中。とりあえずNS社のものを検討中だが，もっといいものがあったら教えてください。（石）

▶私はお笑いが好きだ。毎週の録画はもちろんのこと，ときにはライブにまで足を運ぶ。ちょっと前まではテンションとかモロ諸岡が好きだったけど，最近はフローレンスが気に入っている。ビシバは相変わらず独自路線で好きだし，コロンボのボケぶりも見どころだ。SET隊にも復活してほしいが……。今日はこれからバラライカを見に行く予定。（哲）

▶テレビの健康番組で，胃を悪くすると肩が凝り，膝から下に疲れが溜まるとのこと。いわれてみれば年中肩凝りだし足もだるい。朝起きると吐き気がする。これはいけないと思ってもコーヒーは絶対ブラックだし，カフェイン入りのガムは手放せない。気にすると胃が痛くなるので，結局なにもしないことにした。悟りというやつだな，うんうん。（八）

▼お待たせしているZ-MUSIC BOOKSですが，もうしばらくお待ちください。現在，楽器ADPCM音ディスクの制作，周辺支援ツールの開発を容易にするファンクションの新設……etc.たったひとつの体をこき使ってがんばってます。さらに朗報，あの進藤氏による初心者向けの別冊導入マニュアルがつく予定です。（善）

▶ツァイトのJGフォントをSXにインストールする。むろんシャーペンでもEasydrawでも使える。ふむこれが３次ベジェ曲線か。あくまでデザイン上の好みだが，Windows用ドライバ付属の書体（新明朝/新ゴシック）よりは明朝（細/太）とゴシック（細/中/太）のセットを勧める。ちゃんと揃えると結構な出費。私はフォントコレクタへの道を歩むのか？（A.T.）

▶他人がなにかするとすぐ非難するくせに，自分たちのしていることといったら，すべての事件をすべからく，一杯のかけそば以上に単純化したファンタジーに閉じこめて，理解を超えた物語（ほとんどの物語はヤツらの理解を超えているのだが）を茶飲み話にパッケージ化し続けるだけではないか。水は低いところへ流れるとはよくいったものである。（K）

▶セーラームーンの第２部(TV)が終わった。ありふれた結末のひと言。結局は原作がある程度進行するまでの中継ぎでしかなかったか。矛盾に満ちた設定が議論の対象になり，ストーリー以外で変な盛り上がりを見せたが，離れていったファンも多いと思う。スタッフは自分で自分の首を絞めたのでは。ともかく原作に沿ったちびうさ編には期待大だ。（KO）

▶家から駅までの途中，公園のなかを通る。会社と駅の間も公園を横切ってゆく。昼間に通るときは，わざと遠まわりをしてゆっくりと歩いたりする。お天気の日には，おべんと食べてる女の子やら昼寝のサラリーマンやら犬の散歩やらいろいろいて，ちょっと楽しい。暑い日には子供と一緒に水遊び……はしないけど(……実はしたいんだ)。（ふ）

▶川原氏から行き場を失った増設RAMボードを安く買いたいた。しかし，家のマシンも仕事場のマシンも2Mバイト。プログラミングやお絵描きなど，個人的なこと（しかし，知らぬ間に仕事になっていることが多い）も会社ですますことが多いため，どちらのマシンを増設するかいまだに悩んでいる。ああ，僕の増設RAMボードの幸せはいずこに？（J）

▶以前は「人生の３分の１は中華料理」と宣言していた。これは日本には中華料理店が多いという否定の感情と，自炊も中華風にするとわりにうまくという肯定の感情の両方からである。が，いまは「人生の３分の１はラーメン」という状況になりつつある。なにもこの暑いときにとも思うが，なんとなく食いたくなる。ま，食欲がないよりはマシか。（A）

▶マシン室のLUNAがネットワークにつながった。白いのでartemisと命名されたが，どうも世間の目は冷たい。さて，付録ディスクだが，なんかゲームが多くなりそうな気がする。６月号で制作は続いているとあったポリゴン版MAGICは都合により中断。SION3(仮)は進展なし。ということは……。とにかく今後は要2～4Mバイトなのでよろしく。（U）

▶一般に新聞のパソコン記事はあてにできない。結局，担当記者の力によるんだろうけど，Ｊリーグ情報では東京中日スポーツがやけに冴えてる。戦況分析もいいし，各紙が疑惑扱いしたカズのゴールもオフサイドではないと明示していた(写真が決定的)。加藤久がレッズでなく清水に行くと最初から予想したのも……。（レッズに来てほしかったT）

microOdyssey

世はハードウェア偏重の時代である。少し前にはソフトウェアの重大さということが論じられていたが，いまのパソコン業界ではその部分が薄くなりつつある。

もちろん，本体とシステムソフトウェアとを考えると，いまは本体はどうでもよく，システムソフトウェアがマシンの方向性を決定づける風潮である。ここでいっているのは，むしろ本体，周辺機器を含めたハードウェアと，アプリケーションやゲームなどの市販ソフトウェアのバランスのことである。

パソコンの本体は，ひと頃では考えられなかったような価格になっている。えっ，このスペックでこの値段でいいんですかあ，と少なくとも古くからパソコンに慣れ親しんできたものにとっては，気がひけてしまうような価格の製品が大挙押し寄せているのだ。

そういう状況では，どうしてもユーザーはハードウェアのほうに目がいってしまう。これは否めない事実であろう。高性能(少なくともそう見える)マシンが，10万とか20万円で買えるのであるから，買ってしまった，あるいはどれを買おうかと迷っている人もかなり多いだろう。

で，そちらにお金が割かれた結果，本体を動かすソフトウェアには手が回らないというケースが急増する。そうなると，うちではソフトウェアの数よりもハードウェアのほうの数が多い，とかいうのも，決して笑い話や特殊な状況ではなくなる。

ソフトウェアを作るのが趣味の場合は，世の中の状況がどうあれ自然にこうなるかもしれない。最近なら，パソコン通信や書籍によって手に入るフリーソフトウェアも質，量ともに充実しているから，それだけを楽しんでいるということもあるだろう。

市販ソフトウェアに魅力ある製品が少なくなってきていることも心配の種だ。これではそのものズバリの悪循環になってしまいかねない。全体的なソフトの売り上げを眺めていると，お仕事ソフトはともかく，ゲームソフトまでがII，IIIモノが上位を占めている。昨年よかったソフトはと想像してみても，なかなか思い浮かばない。そのあたりを見ると，X68000はいいソフトがたまに出るだけまだましのようだ。

なんでこういうことを心配するのか，と疑問に思う人もいるだろう。それは単に僕個人がソフトウェアを買わないとパソコンを楽しめない人間だからだ。メインで使っているのは優良ソフトが安くて有名な(?)AMIGAであるが，ソフトウェアだけで，何十万円という金額を投資している。本体にかけている金も大きいが，それを大きく上回っている。

主に使うのはグラフィック関係のソフトだが，たくさんの製品を買ってふるいにかけているわけではなく，同時に何本も使い分けている。それもジャンル分けではなく，同じ3Dソフトでも何本も使うし，ペイントソフトなどもしかり。そのひとつでも欠けると困るのだ。

特殊なものでも，よい製品ならきちんと売れる市場。最低限それだけは維持したいから，いいソフトは購入し，良質のシェアウェアにはちゃんと送金する。周りにソフトウェアが溢れている世界にいるだけに，この心構えは忘れないようにしたい。 (A)

1993年9月号8月18日(水)発売

特集　光学式磁気円盤MO

・大容量リムーバブルメディアを見る

新製品紹介
OS-9/X68030 ver.2.4.5
Oh!X LIVE in'93　銀河鉄道999 ほか
全機種共通システム　SLANG再々掲載

バックナンバー常備店

地域	地区	店名 / 電話
東京	神保町	三省堂神田本店5F 03(3233)3312
	〃	書泉ブックマートBI 03(3294)0011
	〃	書泉グランデ5F 03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン 03(3257)2660
	八重洲	八重洲ブックセンター3F 03(3281)1811
	新宿	紀伊国屋書店本店 03(3354)0131
	高田馬場	未来堂書店 03(3209)0656
	渋谷	大盛堂書店 03(3463)0511
	池袋	旭屋書店池袋店 03(3986)0311
	八王子	くまざわ書店八王子本店 0426(25)1201
神奈川	厚木	有隣堂厚木店 0462(23)4111
	平塚	文教堂四の宮店 0463(54)2880
千葉	柏	新星堂カルチェ5 0471(64)8551
	船橋	リブロ船橋店 0474(25)0111
	〃	芳林堂書店津田沼店 0474(78)3737
	千葉	多田屋千葉セントラルプラザ店 0472(24)1333
埼玉	川越	黒田書店 0492(25)3138
	川口	岩渕書店 0482(52)2190
茨城	水戸	川又書店駅前店 0292(31)0102
大阪	北区	旭屋書店本店 06(313)1191
	都島区	駸々堂京橋店 06(353)2413
京都	中京区	オーム社書店 075(221)0280
愛知	名古屋	三省堂名古屋店 052(562)0077
	〃	パソコンΣ上前津店 052(251)8334
	刈谷	三洋堂書店刈谷店 0566(24)1134
長野	飯田	平安堂飯田店 0265(24)4545
北海道	室蘭	室蘭工業大学生協 0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

基本的に，定期購読に関することは販売局で一括して行っています。住所変更など問題が生じた場合は，Oh!X編集部ではなくソフトバンク販売局へお問い合わせください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

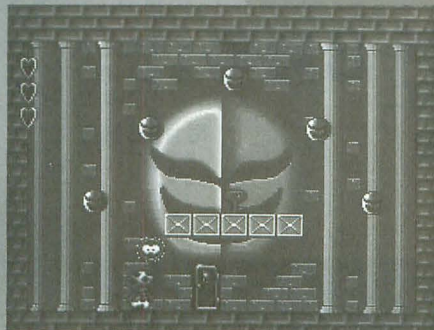
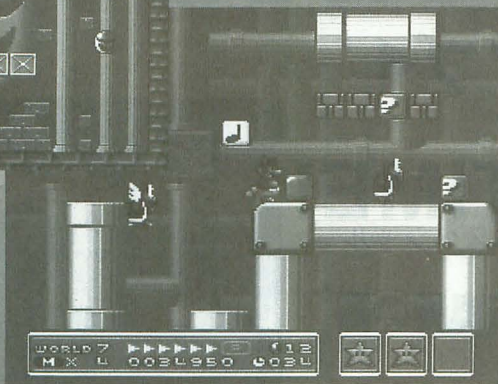
Oh!X 8月号
■1993年8月1日発行　定価600円(本体583円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒103 東京都中央区日本橋浜町3-42-3
Oh!X編集部 ☎03(5642)8122
販売局 ☎03(5642)8100 FAX 03(5641)3424
広告局 ☎03(5642)8111
■印　刷　凸版印刷株式会社

©1993 SOFTBANK CORP. 雑誌 02179-8 本誌からの無断転載を禁じます。
落丁・乱丁の場合はお取り替えいたします。

ご予約・お求めはお近くの本屋さんで

Theスーパーファミコンから
攻略本と別冊を刊行!

Theスーパーファミコン
スーパーガイド・シリーズ
話題の攻略本2冊を7月22日同時発売

スーパーマリオコレクション スーパーガイド
定価950円(税込)

スーパーマリオ1･2･3と
スーパーマリオUSAの
全ステージのマップ付。
完全攻略をめざそう!

ストリートファイターⅡターボ スーパーガイド
定価900円(税込)

全キャラクターの対戦攻略
を完全マスターしよう!

カプコン・ガイドブックの決定版!

7月15日発売

Theスーパーファミコン別冊
ALL ABOUTカプコン
定価1200円(税込)

カプコンのすべて

「ストリートファイターⅡターボ」ほか人気のカプコンSFCゲーム、
FCゲームの総ガイド、カプコンゲーム史、キャラクター図鑑、
裏技大全などカプコンの魅力100%

ソフトバンク出版事業部

SOFT BANK

|スーパーファミコン100%|

7/23号

スーパーファミコン

定価380円(税込)
隔週金曜日発売

全国の書店、コンビニエンスストアにて好評発売中!

ストリートファイターⅡターボ
オールキャラクター完全ガイド

特集　SLG大全［特別編］

箱庭育成型シミュレーションゲームを探る!

悩んでるタール人を救え!

「スーパーファミリーテニス」オールデータガイド

最新作をキャッチアップ!新作FRONT LINE
●聖剣伝説2●ミスティッククエスト●トルネコの大冒険～不思議議のダンジョン
●スーパー信長の野望・全国版●ソード・ワールドSFC

読んで得するスーパーガイド㊢新作SUPER GUIDE
●第3次スーパーロボット大戦●メガロマニア●ワールドヒーローズ●MADARA2

別冊付録
スーパーマリオコレクション
裏ワザ大全集

BEEP! POWERFUL MEGA-MAGAZINE

MEGADRIVE

ビープ!
▶メガドライブ◀

8月号　好評発売中
定価490円
(税込)
毎月8日発売

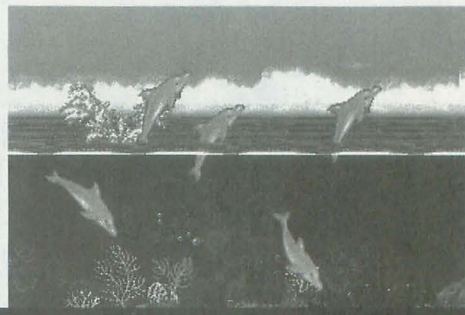
特集　It's SHOW TIME!
'93年東京おもちゃショー&SUMMER CESスペシャル
ピックアップレポート!

特別攻略　ザ・スーパー忍Ⅱ 免許皆伝奥義の書

MEGA-CD PRESS出張版スペシャル
夢見館の物語/AX101/ナイトトラップ/シルフィード

全国600万人の格闘
ゲーマーに捧げるページ　ストⅡダッシュ通信MD

MEGA-CD PRESS
●ソニック・ザ・ヘッジホッグCD●慶応遊撃隊●モンキー・アイランド
●アルシャーク●バリアーム

BEメガ・ホットメニュー
●バーチャレーシング●エコー・ザ・ドルフィン●ファンタシースターⅣ●マクドナルド
●ロケットナイト アドベンチャーズ●ガンスター ヒーローズ●ガントレット

綴じ込み付録SPECIAL
聖魔伝説 3×3EYES
完全攻略
PART
2

SOFT BANK
ソフトバンク出版事業部

お近くの書店でお求めください

microware®

OS-9/X68030 V2.4.5

待望のX68030対応

新世代X68030に、OS-9 V2.4.5が対応。X68030の持つ性能を最大限に引き出し、貴方の夢を創造する力になります。

OS-9は、UNIXライクなリアルタイムOSです。UNIXの長所であるマルチユーザ・マルチタスク、階層ディレクトリ構造および、全てのデバイスをファイルとして扱うユニファイドI/Oなどを備えた、パーソナルコンピュータ向けに洗練されたコンパクトなOSになっています。

■特　長

1.リアルタイム・マルチタスク

メモリサイズの許す限り複数のタスクを起動できます。また、多数のI/Oデバイスがリアルタイム・マルチタスク機能でサポートされています。

2.マルチウィンドウ

オーバラッピングタイプの本格的なマルチウィンドウ『パーソナルウィンドウ』が、サポートされています。

3.マルチユーザ

RS-232C(拡張ボードも対応)に端末接続することにより、最大10ユーザのマルチユーザ環境が使用できます。

4.SCSI

大容量SCSIハードディスクをサポートしています。また、1つのドライブをパーティションで分割使用することで、Human68Kとの混在が可能です。

5.μMACS(マイクロマックス)

μMACSは、UNIX上で広く利用されているスクリーンエディタ"EMACS"のOS-9版サブセットで、テキストファイルを作成したり、変更したりするためのテキストエディタです。

6.日本語処理

日本語フロント・プロセッサVJE-γ V2.0を使用することにより、快適な日本語入力が可能です。

■パッケージ内容

- インスタレーション・マニュアル
- OS-9/X68030本体
  - kernel
  - scf
  - rbf
  - パーソナルウィンドウ
  - プリンタ・ドライバ
  - RS-232Cドライバ
  - SCSIハードディスク・ドライバ
  - フロッピーディスク・ドライバ
  - 日本語処理フロント・エンド・プロセッサVJE-γ
  - shellおよびユーティリティ・コマンド
- オンライン・マニュアル(FD)
  - OS-9ユーザーズ・マニュアル
  - OS-9ユーティリティ・コマンド
  - OS-9μMACS ユーザーズ・マニュアル
  - OS-9VJE-γ ユーザーズ・マニュアル

3.5″/2HD・5″/2HD 各4枚組

〖価格¥**25,000**(税別)〗

X68030対応
Ultra C & Professional Pack,
Technical Tool Kit等、続々登場。

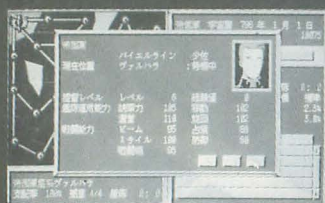
マイクロウェア・システムズ株式会社
〒101 東京都千代田区外神田2丁目17番3号
TEL.03-3257-9000㈹　FAX.03-3257-9200

＊記載の会社名・商品名・名称は、各社の商標又は登録商標です。

SPACE WAR SIMULATION

銀河英雄伝説III

（画面写真はPC98版のものです）

BOTHTEC　イラスト 加藤直之　©QUEST BOTHTEC ©Micro Vision
© 1988 田中芳樹・徳間書店・徳間ジャパン・キティフィルム

銀河英雄伝説III

ブラザーTAKERUスタッフが贈る完成移植版

好評発売中！

X68000版
X68030版対応

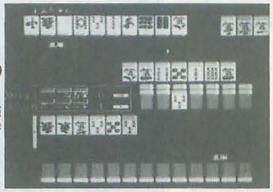
TAKERU価格 ¥7,800（税込）

脳天伝説

究極の二人打ち麻雀ゲームがついに登場！さくさくスピーディーに進むゲーム展開は忙しい現代人にはまさしくうってつけ！　気に入った対戦相手を即座にセレクトでき、麻雀初心者には嬉しい当たり牌表示機能などの親切設計、お楽しみのCGは期待を裏切らない迫力2画面CG！BGMはいまや当然、PCM同期で全16曲！！妥協や手抜きを一切排しエンターティメントを追求したこのソフトを一度お試しください。

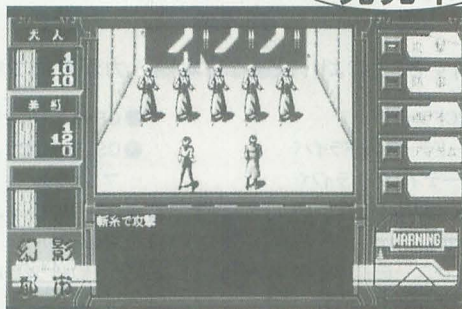
発売中

TAKERU価格 ¥2,500（税込）

■対応機種／X68000版
■制作／IRON GEAR
要マウス、メモリ2Mバイト

ILLUSION CITY 幻影都市

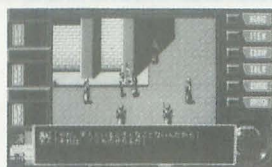
© 1991 MICRO CABIN CORP.

～イリュージョンシティー～

発売中

禍々しき気に満ちた近未来都市、香港。狂気と悪しき欲望とが渦巻くこの都市を、いま一人の男が駆け抜ける。失われた己の過去を求めて、迫り来る危険に自ら身を投じる男、対魔掃討者（デモンバスター）“天人（ダイバー）”は、人民警察（アジターポリス）の対魔特別攻撃班に属する女、“美紅”と共に、その実体さえ知れぬ巨大な悪に対し、渾身の気を込めて愛用の銃（ブラスター）を放つ。果てしなく続く戦いの日々は、いつしか眠ることさえ忘れさせてしまった・・・・。

■8等身キャラクタ採用
■キャラクタ演出革命！！
■ジョイパット＆マウスオペレーション可能
■VRシステムVer.2.5搭載
■MIDI対応
■要2Mバイトメモリー

TAKERU価格 ¥6,800（税込）

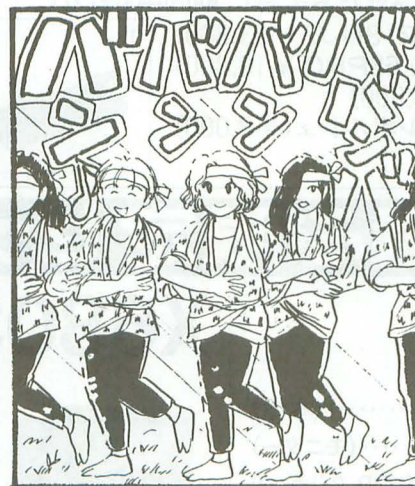
■対応機種／X68000版
■制作／TAKERUソフト
©マイクロキャビン

X68000 ディスカウントキャンペーン 実施中

FSSティグナスの冒険（MNMソフトウェア）2,900円　¥1,200
アルガーナ68（MNMソフトウェア）3,800円……¥1,200
シューティング68KGAMESグランプリ（アモルファス）3,000円…¥1,500
シューティング68KGAMES優秀作2作（アモルファス）3,000円…¥1,500
DINOLAND（ウルフチーム）4,900円……¥2,000
スタートレーダー（TAKERUソフト）4,800円……¥2,000
ロードス島戦記福神漬（ハミングバード）3,500円……¥2,000
NOBLE MIND（アルファシステム）5,900円……¥2,900
シュバルツシルトII（工画堂スタジオ）5,900円…¥2,000
ルーンワース「黒衣の貴公子」（T&Eソフト）6,600円……¥2,000
超人（FIX）4,800円……¥2,000
スーパー上海ドラゴンズアイ（ホットビー）6,200円……¥2,900
オルテウス2（ウインキーソフト）4,800円……¥2,900
マジカルショット（MNMソフトウェア）4,800円…¥2,900
リップスティックアドベンチャー2（フェアリーテール）4,800円　¥3,500
機甲師団（アートディンク）4,800円……¥2,800
ファーサイドムーン（アートディンク）4,800円……¥2,800
栄冠は君に（アートディンク）4,800円……¥2,800
ハイドライドIII（T&Eソフト）4,800円……¥3,800
幻獣鬼（T&Eソフト）5,800円……¥2,000
アクアレス（EXACT）7,000円……¥4,800
A列車で行こうIII（アートディンク）9,800円……¥5,800
チェイスH.Q.（TAKERUソフト）7,800円……¥3,800

ブラザー工業株式会社　〒467 名古屋市瑞穂区苗代町2番1号　TAKERU事務局 (052)824-2493　東京営業所(03)5203-7133　大阪営業所(06)252-4234

通信販売　通信販売をご希望の方は、ソフト名、機種名、住所、氏名、電話番号を明記の上TAKERU事務局まで現金書留でお申し込みください。代金引換は一度、現金書留で申し込んで頂いた方に案内させて頂きます。

満開の電子ちゃん

作・え 岡村祭

講読方法：定期購読もしくはソフトベンダーTAKERUでお買い求めいただけます。
★定期購読の場合＝購読料６ヶ月分6,000円（送料サービス、消費税込）を、
現金書留または郵便振替で下記の宛先へお送り下さい。
現金書留の場合：〒171 東京都豊島区長崎1-28-23 Muse西池袋2F ㈱満開製作所
郵便振替の場合：東京 5－362847 ㈱満開製作所
●ご注文の際は、郵便番号・住所・氏名・電話番号を忘れずに記入して下さい。
●3.5インチディスク版をご希望の方は、「3.5インチ版」とご指定下さい。
●新規購読の方は「新規」と明記して下さい。なお、特に購読開始号のご指定がない場合は既刊の最新号からお送りいたします。
●製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しします。
★TAKERUでお求めの場合＝1部につき1,200円（消費税込）です。
●定期購読版と内容が一部異なる場合があります。御了承下さい。
●お問い合わせ先 TEL(03)3554－9282（月～金 午前11時～午後６時）
（なお、定期購読版のバックナンバーについては定期購読の方のみご注文を承ります）

田中 智（宮城県）

Oh!Xの読者で、電脳倶楽部を知らない人はまず、いないでしょう。そこで、問題なのが、知っているけど、なかなか購読に踏み切れない人が、多いということです。普通なら、ここでスバラシイ内容の数々を説明するところですが、「百聞は一見にしかず」というではありませんか。まず、購読してご覧なさい。損はしません。
昼凜々しい男性諸君！目元涼しい女性諸君！今こそ、電倶の元に集結せよ。この愛のない日本に、諸君らの力で**愛の社会**を築いていこうではありませんか。

P&Aならではの
新品パソコン
5年保証

お支払いは、
便利な商品
到着払い
(手数料要)を
ご利用
下さい。

《業界No.1の"P&Aメンテナンスサポート"》

最高の保証システム

①業界最長の新品パソコン5年保証
(※モニター・プリンター3年間保証!! ※一部商品は除きます。)
②中古パソコンの1年間保証
(モニター・プリンター6ヶ月間保証)
③初期不良交換期間3ヶ月
(※新品商品に限らせていただきます)
④永久買取保証
⑤配達の指定OK!!(土曜・日曜・祭日もOK!!)
⑥夜間配送もOK!!
(※PM6:00〜PM8:00の間　※一部地域は除きます。)

便利でお得な支払いシステム

①翌月一括払い手数料無料(ご利用下さい。)
②業界No.1の低金利
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK!!
⑤84回までの分割、ボーナス併用OK!!
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK!!
⑨現金一括払いOK!!
⑩商品到着払いOK!!(代引き手数料が必要になります。)
(※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。)

●法人向けリースシステム
業務に最適なシステムを構築します。損金処理が可能なリース契約をどうぞ。

注目!! 一括払い手数料(金利)無料(平成5年8月末/9月末/10月末のいずれかをご指定ください。)

SHARP＝X68030エキスパートショップ

P&A＝X68030 ダブルNEWフェア

《7月18日〜8月17日》

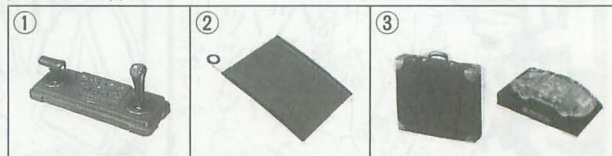
32ビットX68030いよいよ登場。 購入ダブルチャンス!!

X68030発売記念

X68030をモニターとセットで／単品で　購入の方!!

さらに現在お持ちのパソコンと下取り交換されたお客様に期間中もれなく、

①サイバーステック……………………(CZ-8NJ2 ¥23,800)
②CRTフィルター…………………(BF-68PRO ¥19,800)
③X-68000フロッピーアタッシュケース(¥8,000)
とクリスタルポルシェ(¥8,000)

以上のいずれかプレゼント!!

今だからこそ選ぶ限定セット☆☆☆☆☆

X68030(5.25")

●CZ-500CB……………(本体)
●CZ-608DB…………(モニター)
●LHD-FM100E (HDD、ロジテック)
●ハーフ・ハーフケーブル

合計定価 ¥598,600

P&A超特価 **¥406,800**

X68030(3.5")

●CZ-300CB……………(本体)
●CZ-608DB…………(モニター)
●LHD-FM100E (HDD、ロジテック)
●ハーフ・ハーフケーブル

合計定価 ¥588,600

P&A超特価 **¥400,800**

●上記のモニターをCZ-614D-TN(定価¥135,000)に変更の場合¥31,000加算して下さい。
●限定セットにも発売記念サービス品プレゼント中!!

※本広告の掲載の商品の価格については、消費税は含まれておりません。

P&A 全国通販

★頭金なし!!
★即日発送!!

32ビットX68030いよいよ登場(送料¥2,000・消費税別)

5.25"FDD

①CZ-500CB ……………定価¥398,000(本体)
CZ-608D(B) ……………定価¥ 94,800(ディスプレイ)
合計定価¥492,800▶特価TEL下さい。

②CZ-500CB ……………定価¥398,000(本体)
CZ-614DTN ………定価¥135,000(ディスプレイ)
合計定価¥533,000▶特価TEL下さい。

HDDタイプ

③CZ-510CB ………定価¥488,000(本体)(80MBHD内蔵)
CZ-608DB ………定価¥ 94,800(ディスプレイ)
合計定価¥582,800▶特価TEL下さい。

④CZ-510CB ………定価¥488,000(本体)(80MBHD内蔵)
CZ-614DTN ………定価¥135,000(ディスプレイ)
合計定価¥623,000▶特価TEL下さい。

3.5"FDD

①CZ-300CB ……………定価¥388,000(本体)
CZ-608DB ……………定価¥ 94,800(ディスプレイ)
合計定価¥482,800▶特価TEL下さい。

②CZ-300CB ………定価¥388,000(本体)
CZ-614DTN ………定価¥135,000(ディスプレイ)
合計定価¥523,000▶特価TEL下さい。

HDDタイプ

③CZ-310CB ………定価¥478,000(本体)
CZ-608DB ………定価¥ 94,800(ディスプレイ)
合計定価¥572,800▶特価TEL下さい。

④CZ-310CB ………定価¥478,000(本体)
CZ-614DTN ………定価¥135,000(ディスプレイ)
合計定価¥613,000▶特価TEL下さい。

※お支払いは、便利な商品到着払い(手数料要)をご利用下さい。

旧シリーズ今が買いどき!! (クレジット表:送料、消費税込み)
X68000 Compact XVI/XVI 送料¥2,000、消費税別

Compact XVI

① ●CZ-674C-H(本体)
●CZ-608D-H(モニター)
定価¥392,800
P&A超特価¥193,000

12回	17,600	24回	9,300	36回	6,500	48回	5,100	60回	4,300

XVI

① ●CZ-634C-TN(本体)
●CZ-608D-H(モニター)
定価¥462,800
P&A超特価¥213,000

12回	19,500	24回	10,300	36回	7,100	48回	5,600	60回	4,700

上記のモニターをCZ-614Dに変更

② ●CZ-674C-H(本体)
●CZ-614D-TN(モニター)
●CZ-6CR1(RGBケーブル)
●CZ-6CT1(TVコントロール)
定価¥443,000
P&A超特価¥233,000

12回	21,200	24回	11,200	36回	7,800	48回	6,100	60回	5,200

上記のモニターをCZ-614Dに変更

② ●CZ-634C-TN(本体)
●CZ-614D-TN(モニター)
定価¥503,000
P&A超特価¥243,000

12回	22,200	24回	11,700	36回	8,100	48回	6,400	60回	5,300

※上記(1)のモニターをCZ-607D-TN(定価¥99,800)に変更の場合¥18,000加算して下さい。
※ディスケット10枚、ゲームソフト1ヶプレゼント。

X68000シリーズ~P&Aスペシャルセット(送料¥2,000・消費税別)

SUPER-HD ★ハードディスク81MB搭載!! ※ディスケット10枚・ゲームソフト1ヶプレゼント

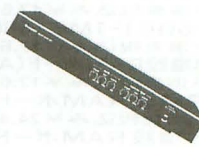
Ⓐセット:■CZ-623C-TN(単品) …………定価¥498,000▶特価¥158,000
Ⓑセット:■CZ-623C-TN+CZ-606D ……定価¥577,800▶特価¥213,000
Ⓒセット:■CZ-623C-TN+CZ-608D ……定価¥592,800▶特価¥226,000
Ⓓセット:■CZ-623C-TN+CZ-607D ……定価¥597,800▶特価¥228,000
Ⓔセット:■CZ-623C-TN+CZ-614D ……定価¥633,000▶特価¥248,000
Ⓕセット:■CZ-623C-TN+CU-21HD ……定価¥646,000▶特価¥258,000

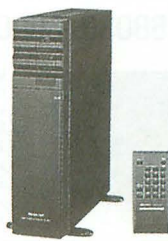
周辺機器特選品コーナー

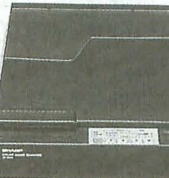
JX-325X
カラーイメージスキャナ

定価¥190,000
特価¥138,000

JX-32F12(写真上部分)
定価¥148,000
特価¥107,000

CZ-6VTI
カラーイメージユニット

定価¥69,800
特価¥49,500

CZ-6TU
RGBシステムチューナー

定価¥33,100
特価¥23,900

JX-220X
カラーイメージスキャナ

定価¥168,000
特価¥121,000

CZ-8NSI
カラーイメージスキャナ

定価¥188,000
特価¥133,000

(X68030用)増設RAMボード&数値演算プロセッサ

CZ-5BE4
定価¥54,800
特価¥42,000

CZ-5ME4
定価¥49,800
特価¥38,000

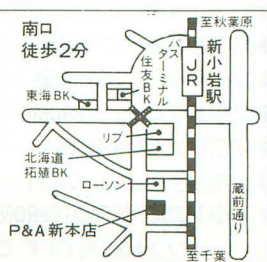
CZ-5MPI
定価¥54,800
特価¥42,000

〔銀行振込でお申し込みの方〕(電信扱いでお振込み下さい。)

〔振込先〕さくら銀行 新小岩支店
当座預金 2408626
(株)ピー・アンド・エー

超低金利クレジット率

回数	3	6	10	12	15
手数料	2.9	3.9	4.9	5.4	8.4
回数	24	36	48	60	72
手数料	11.4	15.9	20.9	26.9	34.9

P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目2番地20号
☎03-3651-0148(代) FAX.03-3651-0141
●営業時間:AM10:00~PM7:00
日・祭:AM10:00~PM6:00
●定休日/毎週水曜日

●価格は流通事情により変動致しますので、銀行振込・書留等の送付前にあらかじめお電話にてご確認下さい。

※お支払いは、便利な商品到着払い(手数料 要)をご利用下さい。

P&Aならではの 新品パソコン 5年保証

《業界No.1の"P&Aメンテナンスサポート》

最高の保証システム

①業界最長の新品パソコン5年保証
(※モニター・プリンター3年間保証!! ※一部商品は除きます。)
②中古パソコンの1年間保証
(モニター・プリンター6ヶ月間保証)
③初期不良交換期間3ヶ月
(※新品商品に限らせていただきます)
④永久買取保証
⑤配達の指定OK!!(土曜・日曜・祭日もOK!!)
⑥夜間配送もOK!!
(※PM6:00〜PM8:00の間　※一部地域は除きます。)

便利でお得な支払いシステム

①翌月一括払い手数料無料(ご利用下さい。)
②業界No.1の低金利
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK!!
⑤84回までの分割、ボーナス併用OK!!
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK!!
⑨現金一括払いOK!!
⑩商品到着払いOK!!(代引き手数料が必要になります。)

(※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。)

モデム(送料¥1,000 消費税別)

■FMMD-311G
(富士通)定価¥35,800
▶特価¥24,800
(送料・消費税込み¥26,574)

■PV-M24V5
(AIWA)定価¥36,800
▶特価¥25,700
(送料・消費税込み¥27,501)

■MD-24FB5V
(オムロン)定価¥39,800
▶特価¥23,500
(送料・消費税込み¥25,235)

注目!!一括払い手数料(金利)無料(平成5年8月末/9月末/10月末のいずれかをご指定ください。)

●お近くの方は、お立寄下さい。専門係員が説明いたします。
●本体単品でも受付します。詳しくは、お電話にてお問合せ下さい。

周辺機器コーナー (送料¥1,000・消費税別)

No.	品名	定価	特価
①	BF-68PRO	¥19,800	¥14,400
②	CZ-8NM3	¥9,800	¥7,200
③	CZ-8NT1	¥13,800	¥10,000
④	CZ-6BE2A	¥59,800	¥42,800
⑤	CZ-6BE2B	¥54,800	¥39,300
⑥	CZ-6BE2D	¥54,800	¥39,300
⑦	CZ-6BF1	¥49,800	¥35,800
⑧	CZ-6BP1	¥79,800	¥57,000
⑨	CZ-6BM1	¥26,800	¥19,300
⑩	AN-S100	¥36,600	¥26,300
⑪	CZ-6SD1	¥44,800	¥32,500
⑫	CZ-6BN1	¥29,800	¥21,500
⑬	CZ-6BV1	¥21,000	¥15,200
⑭	CZ-6BC1	¥79,800	¥57,000
⑮	CZ-6BG1	¥59,800	¥43,000
⑯	CZ-6BU1	¥39,800	¥28,500
⑰	CZ-6PV1	¥198,000	¥142,000
⑱	CZ-6BS1	¥29,800	¥21,500
⑲	CZ-8NJ2	¥23,800	¥17,500
⑳	CZ-6BL2	¥298,000	¥214,000
㉑	CZ-6CSI(674C用)	¥12,000	¥8,900
㉒	CZ-68HA		¥91,000
㉓	CZ-6CR1(RGBケーブル)	¥4,500	¥3,600
㉔	CZ6CT1(テレビコントロール)	¥5,500	¥4,400
㉕	CZ-6BP2	¥45,800	¥33,300

■SX-68MII(MIDI)(サコム)
定価¥19,800▶特価¥13,500
(送料・消費税込み¥14,935)

FDD(5インチ×2基)
■CZ-6FD5
(シャープ)
(定価¥99,800)
P&A超特価
¥49,800

プリンター
(ケーブル用紙付・送料¥1,000・消費税別)

■CZ-8PC5-BK
定価¥96,800
特価¥68,500

■CZ-8PK10
定価¥97,800
特価¥71,000

カラーイメージジェット

■IO-735X-B
定価¥248,000
特価¥135,000
(送料・消費税込み¥140,080)

X68000/68030 専用ハードディスク (送料¥1,000 消費税別)

外付

■富士通(純正)
⊙FMHD-1201G
(120MB、17ms、ケーブル付)
定価¥70,000
P&A特価¥49,800

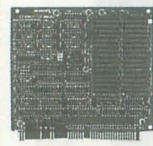
■システムサコム
⊙HD-K200A(モッキンバード)
(200MB、17ms)
定価¥79,800
P&A特価¥61,000

〈ロジテック〉
①LHD-FM100E(定価¥99,800)
▶P&A特価¥TEL下さい。
②LHD-FM200E(定価¥138,000)
▶P&A特価¥66,000

〈ジェフ〉
③GF-120(定価¥108,000)·▶特価¥49,800
④GF-200(定価¥138,000)·▶特価¥62,000
⑤GF-240(定価¥148,000)·▶特価¥89,000

内蔵

CZ-500C/300C専用
●CZ-5H08(80MB/23ms)
定価¥98,000▶特価¥71,800
●CZ-5H16(160MB/18ms)
定価¥135,000▶特価¥99,500

X68030/68000 メモリボード (I/Oデータ)

①SH-5BE4-8M(X68030用)
(送料・消費税込み¥47,586)特価¥45,500
②SH-6BE1-1ME(600C専用)
(送料・消費税込み¥12,669)特価¥11,600
③1MB増設RAMボード(ACE/PRO/PROII用)
(送料・消費税込み¥12,669)特価¥11,600
④2MB増設RAMボード(拡張スロット用)
(送料・消費税込み¥24,411)特価¥23,000
⑤4MB増設RAMボード(拡張スロット用)
(送料・消費税込み¥40,170)特価¥38,300

P&A特選パソコンラック&OAチェアー (消費税込み)(送料無料、離島を除く)

①3段 ¥8,240
②4段 ¥9,785
③5段 ¥11,845

700　350　160　415　60　330　570　1,260

※全機種→キャスター付
※上から2番目棚板移動可能(4/5段)
※フレーム色:
4段→黒、3/5段→ホワイト(W-640)

①¥11,845
②¥20,394

X68000用ソフトコーナー (送料¥700・消費税別)

品名	定価	特価
◆Z'sSTAFFPRO68KVer.3.0(ツアイト)	¥58,000	¥37,500
◆Z'sTRIPHONYデジタルクラフト(ツアイト)	¥39,800	¥27,000
◆テラッツォ(ハミングバード)	¥19,400	¥13,600
◆ラジックパレット(ミュージカルプラン)	¥19,800	¥14,200
◆たーみのる2(SPS)	¥17,800	¥13,000
◆Mu-1Super	¥39,800	¥28,500
◆CMA68K(シティソフト)	¥29,000	¥21,800
◆サイクロンEXPRESSα68	¥98,000	¥69,000
◆C-TRACE68Ver.3.0(キャスト)	¥98,000	¥68,500
◆C&ProfessionalPackV3.2(マイクロウェアジャパン)	¥80,000	¥57,800
◆ウエットペイント1〜3(ウエーブトレイン)〔各〕	¥15,000	¥11,500
◆マチエール(サンワード)	¥39,800	¥28,800
◆WindexPRO68(JEL)	¥28,000	¥20,500
◆CZ-213MSDMUSICPRO68K	¥18,800	¥13,200
◆CZ-214MSDSOUNDPRO68K	¥15,800	¥11,300
◆CZ-215MSDSamplingPRO68K	¥17,800	¥12,500
◆CZ-220BSDDATAPRO68K	¥58,000	¥40,000
◆CZ-225BSV Multiword Ver.1.1	¥32,000	¥23,000
◆CZ-243BSDCYBERNOTEPRO68K	¥19,800	¥15,000
◆CZ-247MSDMUSICPRO68K(MID)	¥28,800	¥20,500
◆CZ-249GSDCANVASPRO68K	¥29,800	¥22,000
◆CZ-251BSDHyperword	¥39,800	¥29,400
◆CZ-253BSDCARDPRO68KVer.2.0	¥29,800	¥22,700
◆CZ-257CSDCommunicationPRO68KVer.2.0	¥19,800	¥15,300
◆CZ-258BSDTeleportionPRO68K	¥22,800	¥16,900
◆CZ-261MSDMUSICstudioPRO68KVer.2.0	¥28,800	¥21,200
◆CZ-263GWDEasypaintSX-68K	¥12,800	¥9,800
◆CZ-265HSDNewPrintShopVer.2.0	¥20,000	¥15,400
◆CZ-266BSDPressConductorPRO68K	¥28,800	¥22,000
◆CZ-267BSDCHARTPRO68K	¥38,000	¥29,800
◆CZ-272CWCCommunicationSX68K	¥19,800	¥14,500
◆CZ-275MWDSOUNDSX68K	¥15,800	¥11,500
◆CZ-284SSDOS-9/X68000Ver.2.4	¥35,800	¥25,600
◆CZ-285LSDC-CompilerPRO68KVer.2.1	¥44,800	¥32,500
◆CZ-286BSDBUSINESSPRO68KPopular	¥28,000	¥20,500
◆CZ-290TWD SX-WINDOW ディスクアクセサリー集	¥14,800	¥11,500
◆CZ-294SS(5")/SSC(3.5") SX-WINDOW Ver.3.0	¥19,800	¥15,200
◆CZ-288LWD開発キット(workroom)	¥39,800	¥29,700

☆ゲームソフト25%OFF OK!!(一部ソフト除く)

注目!!
★中古パソコン1年間保証システム!!
（※モニター、プリンター6ヶ月間保証）

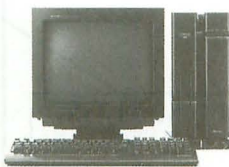
中古その場で高価現金買取り 高価下取りOK!! 電話一本ですぐ買える!

中古パソコンはP&Aにおまかせ!!

※お支払いは、便利な商品到着払い(手数料要)をご利用下さい。

P&A特選今月の中古特選品

- CZ-600C…………¥55,000
- CZ-601C…………¥65,000
- CZ-611C…………¥70,000
- CZ-652C…………¥75,000
- CZ-612C…………¥95,000
- CZ-603C…………¥85,000
- CZ-653C…………¥78,000
- CZ-612C ………¥ 90,000
- CZ-623C ………¥110,000
- CZ-674C ………¥108,000
- CZ-634C ………¥130,000
- CZ-644C ………¥178,000

（上記は単品価格、モニター別売）

新古品 限定
- CZ-674CH
- CZ-608DH

¥168,000

中古品
- CZ-674CH
- 68000専用モニター付

¥138,000

限定
- CZ-634CTN(チタン)(中古)
- CZ-613D(グレー)(新品)

¥200,000

(モニターをCZ-614DTN(チタン)に変更の場合¥20,000加算)

中古品
- CZ-634CTN
- 68000専用モニター付

¥163,000

新古品 限定
- CZ-644CTN
- CZ-604DB

¥248,000

中古品
- CZ-644CTN
- 68000専用モニター付

¥213,000

グレードアップ

現在お持ちのパソコンとX68030シリーズを下取り交換されたお客様に期間中もれなく!

①サイバーステック (CZ-8NJ2¥23,800)
②CRTフィルター (BF-68PRO¥19,800)
③X-68000フロッピーアタッシュケース (¥8,000)
とクリスタルポルシェ(¥8,000)

以上のいずれかプレゼント!!

①
②
③
グレードアップ差額表

下取 ＼ 新品	CZ-500CB	(80MB HD内蔵) CZ-510CB	CZ-300CB	CZ-310CB
CZ-674C	¥195,000	¥263,000	¥190,000	¥250,000
634C	¥175,000	¥243,000	¥170,000	¥230,000
644C	¥125,000	¥193,000	¥120,000	¥180,000
623C	¥205,000	¥273,000	¥200,000	¥260,000
653C	¥255,000	¥323,000	¥240,000	¥300,000
604C	¥225,000	¥293,000	¥230,000	¥290,000
603C	¥255,000	¥323,000	¥250,000	¥310,000
602C	¥255,000	¥323,000	¥250,000	¥310,000
601C	¥265,000	¥333,000	¥250,000	¥310,000
600C	¥275,000	¥343,000	¥260,000	¥320,000
611C	¥255,000	¥323,000	¥240,000	¥300,000
612C	¥245,000	¥313,000	¥240,000	¥300,000
613C	¥235,000	¥303,000	¥240,000	¥300,000
PC-9801RX2	¥245,000	¥313,000	¥240,000	¥300,000
DA2	¥215,000	¥283,000	¥210,000	¥270,000

中古・高価現金買取り/下取りOK!!

■まずはお電話下さい。
下取り専用 買取り電話▶ ☎03-3651-1884 FAX. 03-3651-0141
■下取り・買取りで、お急ぎの方は、直接当社に来店、または宅急便にてお送りください。

買取り価格…完動品・箱/マニュアル/付属品の価格です。

- 下取りの場合…価格は常に変動していますので査定額を電話で確認してください。（差額は、P&A超低金利クレジットをご利用ください。）
- 買取りの場合…現品が着き次第、2日以内に高価買取金額を連絡し、振込み、又は書留でお送り致します。
- 近郊の方はP&A本店に直接お持ちください。即金にて¥1,000,000までお支払い致します。

- 最新の在庫情報・価格はお電話にてお問い合せください。
- 買い取りのみ、または、中古品どうしの交換も致します。詳しくは電話にて、お問い合せください。
- 価格は変動する場合もございますので、ご注文の際には必ず在庫をご確認ください。
- 本商品の掲載の商品の価格については、消費税は、含まれておりません。
- 現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せください。

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
- 商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。(プリンター・プロッピーの場合、本体使用機種名をご明記のこと)

〔銀行振込でお申し込みの方〕
- 銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。
(電信扱いでお振込みください。)

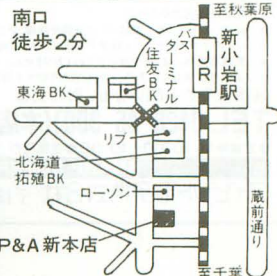
〔振込先〕さくら銀行 新小岩支店
当座預金 2408626 (株)ピー・アンド・エー

〔クレジットでお申し込みの方〕
- 電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
- 現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
- 1回～84回払いまで出来ます。但し、1回のお支払い額は¥1000円以上。

《便利な超低金利クレジットをご利用ください》

- 月々¥1,000円からOK!!
- ボーナス払いOK!!(夏冬10回までOK)
- 支払い回数1回～84回
- お払いは、8ヶ月先からでもOK!!

超低金利クレジット率

回数	3	6	10	12	15	24	36	48	60	72
手数料	2.9	3.9	4.9	5.4	8.4	11.4	15.9	20.9	26.9	34.9

株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目2番地20号
☎03-3651-0148(代) FAX. 03-3651-0141

営業時間
平日:AM10:00～PM7:00
日祭:AM10:00～PM6:00

- 現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込みください。詳しくは、お電話でお問い合わせください。

近日発売予定

Mu-1 GS

要2MB

■ローランド社SC-55mk-II *SOUNDCanvas*対応/MIDIマルチレコーダー

標準価格 ¥28,000（税抜き）

☆使いやすくなったGS音源エディット※
☆RS-232C/MIDI出力対応
（注意:出力のみ対応、単独使用不可/要MIDIボード）
☆簡単エクスクルーシブ入力
☆シーケンス機能はMu-1 Super
☆スタンダードMIDIファイル対応
☆ミュージ郎 II データコンバート機能追加
☆国本佳宏/GS対応デモ曲収録

※Mu-1, Mu-1 Superのユーザーの方々には、
バージョンアップのご案内をお送りいたします。

※下記コントロールコードのリアルタイムエディットおよび任意の位置へのステップ入力が可能
◎リバーブ、コーラスのセンドレベル、パンポット、バンク、レベル、キーシフト
◎TVFカットオフフレクェンシー、TVFレゾナンス、TVF&TVA・アタック、ディケイ、リリース・タイム
◎ドラムインストゥルメント・ピッチ、リバーブセンド、コーラスセンド、パンポット、ボリューム

MIDIボード付/Mu-1 GS
通販のみ限定100セット発売
構成■システムサコムSX-68M
（旧バージョン）
■オーディオ拡張キット
■Mu-1 GS
特別価格¥32,000(送料・税込)

通信販売の方法：現金書留にて右記の宛先
"Mu-1通販係"まで代金をお送りください。
必ず、住所、氏名、電話番号を記入してください。

〒213 神奈川県川崎市高津区下作延1043
株式会社 サンワード
TEL 044-855-4335

お待たせしました!!
オーディオ拡張キット(SX-68M II用) ¥8,000(送料・税込)
※スロットカバー は黒のみ
の通信販売を開始します。

SHARP X68030
32bit PERSONAL WORKSTATION

AIBIT

7/15〜8/15

X68030祭り・下取りセール

お買上げ機種	下取り機種	
CZ-500C	CZ-600C	¥269,000
CZ-510C	CZ-600C	¥338,000
CZ-300C	CZ-600C	¥267,000
CZ-310C	CZ-600C	¥330,000

在庫処分
CZ-634C X68000XVI ⇨ ¥168,000
CZ-674C X68000COMPACT ⇨ ¥148,000

在庫処分
コンパクトフロッピーディスクユニット(2D)
無料プレゼント
CZ-300F
送料は着払いにてお願い致します。
※詳しい問い合せはTelにて.//

書院パソコン
アイビット特価

PC-WD1A	標準価格330,000円
PC-WD1AD	標準価格450,000円
(40MBハードディスク内蔵タイプ)	
PC-WD1B	標準価格430,000円
PC-WD1BD	標準価格590,000円
(80MBハードディスク内蔵タイプ)	

〈全商品新品完全保証付〉
★シャープ・シャープ周辺機器（拡張機器全機種、プリンター他）・富士通・NEC取り扱い。
★シャープ・カシオポケコン全機種取り扱い。PACIFIC・YHP・キャノンも取り扱い。
★上記商品価格には、消費税は含まれておりません。
★特価表及び資料をご希望の方は、200円切手を同封の上お送りください。
通信販売のお問い合せ、御注文は
TEL.0426-45-3001(本店) FAX.0426-44-6002
●営業時間/10:00〜19:00●電話受付/9:00〜21:00 迄可●定休日/水曜日
SHARP SUPER EXE SHOP
アイビット電子株式会社 〒192 東京都八王子市北野町560-5

上記の広告商品は店頭販売もしております。
全通販 国信売
北海道から沖縄まで
★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。
富士銀行八王子支店 (普)1752505

SHARP SHARP SHARP SHARP SHARP SHARP SHARP

激安!! 夏のお買得市 激安!!

X68030
32bit PERSONAL WORKSTATION

ピュア32ビットMC68EC030搭載。クリエイティブパワーが花ひらくX68030シリーズ

X68030
本体+キーボード+マウス・トラックボール
5.25インチFDDタイプ
CZ-500C-B(チタンブラック)標準価格398,000円(税別)
HDタイプ
CZ-510C-B(チタンブラック)標準価格488,000円(税別)
14型カラーディスプレイ(ドットピッチ0.28㎜)
CZ-608D-B(チタンブラック)
標準価格94,800円(税別・チルトスタンド同梱)

シスペック特価

X68030 Compact
本体+キーボード+マウス
2DD対応3.5インチFDDタイプ
CZ-300C-B(チタンブラック)
標準価格388,000円(税別)
HDタイプ
CZ-310C-B(チタンブラック)
標準価格478,000円(税別)
14型カラーディスプレイ(ドットピッチ0.28㎜)
CZ-608D-B(チタンブラック)
標準価格94,800円(税別・チルトスタンド同梱)

シスペック特価

お買得コーナー

X68000
PERSONAL WORKSTATION・XVI

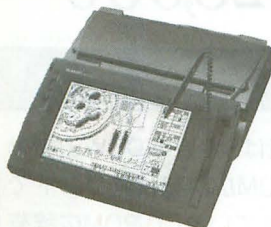
なか身は32ビット

プロセッサの未来を先取、洗練されたアーキテクチャを誇るMPU MC68000シリーズを搭載。

X68000 XVI Compact
本体+キーボード+マウス、3.5インチFDDタイプ、CZ-674C 定価¥298,000
お買得特価¥138,000
月々¥4,740×36回 ㊍なし
モニターとセットなら更にお買得!!

書院パソコン
このパソコンには、ワープロ機能がついています。ハードディスク内蔵(Dタイプ)、OADG仕様、DOS/V対応(別売)。先の先まで考えているから、「国際標準」をキーワードにしました。

PC-WD1A 定価¥330,000
お買得特価¥198,000
月々¥6,800×36回 ㊍なし
PC-WD1AD
定価¥450,000(40MB HDD内蔵タイプ)
お買得特価¥258,000
月々¥8,860×36回 ㊍なし

大人気ワープロ

ペン1本で、手書き編集から手書き入力、イラスト作成、データ管理まで……。使いやすさと表現力をさらに高めた、ハイグレードワープロ。
WD-A770 定価¥240,000
シスペック特価

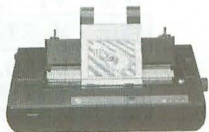
X68000シリーズ専用ハードディスク

Corsair
LHD-FM200E
定価¥138,000
お買得特価¥74,800
LHD-FM100E
定価¥99,800
お買得特価¥47,800

Eclace エクラス
MOディスクユニット
LMO-FMX330
定価¥178,000
お買得特価
¥138,000

スキャナー

600DPI、1,677万色、高品位、高画質、高速読み取りを実現。
カラーイメージスキャナ
JX-325X
定価¥190,000
シスペック特価

プリンター

3種類の制御コマンドを搭載。質感鮮やか、高品位カラーイメージジェット。
IO-735X-B
定価¥248,000
シスペック特価

モデム
アイワ パソコン通信用モデム
PV-AF144V5
定価¥64,800
シスペック特価

新製品
液晶表示機能。G3高速FAXも搭載した14400bps超高速モデム。

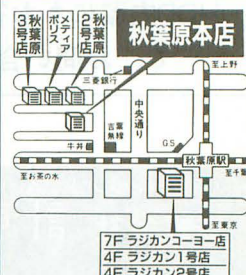
アイワ	PV-AF24V5	¥29,800	シスペック特価
	PV-PF24	¥26,800	シスペック特価
オムロン	MD-24FL10V	¥21,800	シスペック特価
	MD-24XL10V	¥26,800	シスペック特価
	MD-96XL10V	¥46,800	シスペック特価
マイクロコアー	MC-24PA5	¥19,800	特価¥14,800
	MC-24FA5P	¥24,800	特価¥21,000

通信販売のお申し込み、お問い合わせは 東京03-3257-0281(代)

★シスペック通信販売部
〒101 東京都千代田区外神田1-8-11(本店3F)
★銀行振込
三菱銀行 秋葉原支店 普通No.4696203 シスペック㈱

シスペック㈱本店 ☎03-3257-6337
〒101 千代田区外神田1-8-11
年中無休 営業時間AM10:00〜PM7:00

★広告に載っている価格には消費税は含まれていません。クレジット例には消費税が含まれています。
★パソコン・ワープロ・ファクシミリ・OA …月々¥3,000からの超低金利クレジット!!
★さらに有効に!!ご購入からご使用まで、あらゆるご相談に応じます。

シスペック秋葉原各店ご案内

マルチメディア
新世界への誘い
CD-ROM for X68000

X PRO SHOP
BASIC HOUSE
KEISOKUGIKEN Corp.
TEL0286-22-9811 FAX25-3970

倍速CDROM-DRIVE KGU-XCDII

最速200msec 256kキャッシュ

ご好評をいただいておりますX68000用CD-ROM DRIVE KGU-XCDが、新しくなりました。使用ドライブを従来の東芝XM-3301からXM-3401に変更。より速いファイルリードが可能になりました。XM-3401は平均シークタイム200m秒、256Kbytesにも及ぶ大容量キャッシュ機能や倍速回転による高スループット等により最高速の実力です。

※現バージョンのCDROMドライバはHuman68k Ver.3.0では動作しません。近日中に対応する予定です。

PhotoCD™

PhotoCDはコダック社とフィリップス社の共同開発で世に放たれた全く新しい写真の保存形態です。一般的に撮影された写真を安価にCD-ROMに書き込み、必要に応じていつでも閲覧できます。

X68000＆-KGU-XCDでの対応を予定しております。

CD-ROM soft第一弾

Free Software Selection

価格¥5,000-

中身は買ってからのお楽しみ、CD-ROMならではの大容量での内容です。

KGU-XCD II

標準価格*128,000-*

CZ-634C(XVI)大特価!!

当社在庫限り!!

***¥184,000*(税別)**

＊在庫状況をお確かめください。

*CZ-614D(ディスプレイTV)とセット

***¥284,000*(税別)**

*CZ-608D(ディスプレイ)とセット

***¥259,000*(税別)**

……その他、ご要望により大容量HDD内蔵にもお答えいたしますので、お問い合わせください

KGB-X68PRKII値下げ!!

コプロ無しモデル

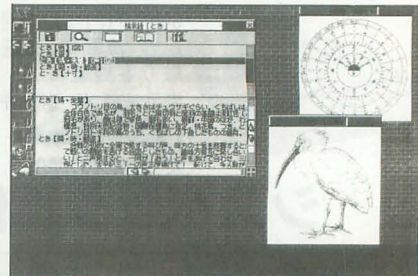
型番	定価	特価
PRKII-02	¥ 55,000	¥ 41,250
PRKII-04	¥ 90,000	¥ 63,000
PRKII-06	¥125,000	¥ 87,000
PRKII-08	¥160,000	¥112,000

コプロ付きモデル

型番	定価	特価
PRKII-12	¥ 85,000	¥ 63,750
PRKII-14	¥120,000	¥ 84,000
PRKII-16	¥155,000	¥108,500
PRKII-18	¥190,000	¥133,000

※CZ-500/300シリーズでのご使用はPRK-08のみ対応となります。
※メインメモリ標準1MBの機構では、専用増設1MBメモリが必要です。

SX-広辞苑

SX-広辞苑はSX-WINDOW上で動作するCD-ROM広辞苑検索ソフトです。市販されているCD-ROM広辞苑第三版を検索できます。

SX-広辞苑（ソフトのみ）
¥19,800

SX-広辞苑 CD-ROM広辞苑セット
¥45,000

低金利クレジット　通信販売送料 全国一律¥1,000 長期クレジット可能

※表示価格に消費税は含まれておりません

株式会社 計測技研　マイコンショップBASIC HOUSE
本社／ショールーム／通販部
〒321 栃木県宇都宮市竹林町503-1
TEL 0286-22-9811　FAX 0286-25-3970

パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

第7回 Percomboyさん ID:JH241302

今回は、昨年10月にX68000のユーザーになられたばかりのPercomboyさんの登場です。まだ使用期間は半年あまりだというのに、すでに立派なX68000フリークのご様子。それもこれも、パソコン通信というメディアがサポートしているから(?)かもしれません。Percomboyさんのx68000への熱いメッセージをどうぞ。

＝基本データ＝

■使用機種名：X68000ACE-HD(CZ-611C-BK)
■主な所有周辺機器：プリンタ(TX-24CL：スター)
モデム(SPORTSTER14,400Fax：USRobotics)
外部ターミナル・サブマシン
(CZ-881C〈X1turboZII〉：シャープ)
■使用開始時期：1992年10月

■X68000を選んだ理由は？

思想、機能を見て、夢と希望の残った唯一のマシンだと思ったから。

■主にどんな用途に使われていますか？

ゲームとプログラミング…のはずだったのだが、「主に」は、通信と音楽と各種文書作成。

■お気に入りのゲームソフトは？

「トンネルズ&トロールズ　カザンの戦士たち」(スタークラフト)。コンピュータRPGで、ある程度「テーブルトーク」の雰囲気を出しているところがいい。内容について知りたい方は、J&P HOTLINE内のSIG「SFXぱらだいす」#5、「エンコム社・会議室」をどうぞ。

■X68000のよいところ、楽しい部分など。

SX-WINDOW（X68000独自のウインドウシステム）が、10MHzのマシンでも満足なスピードで動くところ。でも、RAMが2Mでは、少し苦しいですが…。

■X68000、こうだったらいいのになぁと思うことは？

世間から、「正当な評価」を受けられればなぁと思います。条件はいろいろつくけれど、スピードも、初代・10MHz機でも決して遅くないし。フリーソフトでなら良いソフトはたくさんあるし。OSなどの基本ソフトは、他のマシンよりも割安だし。ハードの値段も中古ならばAT互換機よりも安く手に入るし。あとは、性能面からいっても、高機能でプログラムしやすいと思う。

■J&P HOTLINEに入会したきっかけはなんですか？

友人に「通信」を勧められて。彼がここに来ていたから。

■オンラインであなたの常駐コーナー(好きなメニュー)はどこですか？

SIG（CZ-CLUB、Dreamy Booksl、FUTURE FORUM、SFXぱらだいす、SHARP-HOTLINE）

■X・MZユーザーに知らせたいHOTLINEのコーナーはどこですか？

MZユーザーなら「SHARP-HOTLINE」、Xユーザーにはそれに加えて「CZ-CLUB」

■あなたにとって、J&P HOTLINEとは？

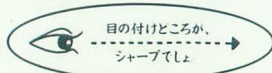
私にとって、J&P HOTLINEは“中央ロビー”だ。なぜなら、「いろんな人が出入りし、出会いもあり、時には待ち合わせ場所にすることもある」ということ。

J&P HOT LINEへのご入会はスタータキットで。

お求めは、下記のお店へ。又は現金書留にて、¥3,000＋¥90(消費税3%)＝¥3,090を事務局までお送り下さい。
すぐにスタータキットをお送りします。

お問い合わせは――
〒556 大阪市浪速区日本橋西1-6-5 上新電機株式会社
J&P HOTLINE事務局宛　TEL.(06)632-2521

スタータキットのお求めはJ&P各店でどうぞ。

店名	電話	店名	電話	店名	電話	店名	電話	店名	電話	店名	電話
										和歌山店	☎(0734)28-1441
渋谷店	☎(03)3496-4141	津田沼店	☎(0474)72-5211	大須店	☎(052)262-1141	くずは店	☎(0720)56-8181	さんのみや1ばん館	☎(078)231-2111	和歌山南店	☎(0734)25-1414
町田店	☎(0427)23-1313	越谷店	☎(0489)66-1221	テクノランド	☎(06) 634-1211	千里中央店	☎(06) 834-4141	西宮店	☎(0798)71-1171	学園前店	☎(0742)49-1411
八王子店	☎(0426)26-4141	焼津インター店	☎(054)626-3311	メディアランド	☎(06) 634-1511	摂津富田店	☎(0726)93-7521	伊丹店	☎(0727)77-5101	奈良1ばん館	☎(0742)27-1111
立川店	☎(0425)36-4141	にいがた1ばん館	☎(025)241-3711	コスモランド	☎(06) 634-3111	寝屋川店	☎(0720)34-1166	姫路店	☎(0792)22-1221	新大宮店	☎(0742)35-2611
三鷹店	☎(0422)31-6251	富山店	☎(0764)22-5033	U.S.LAND	☎(06) 634-1411	枚方バイパス店	☎(0720)48-1211	京都寺町店	☎(075)341-4411	郡山インター店	☎(07435)9-2221
横浜店	☎(045)313-6711	金沢店	☎(0762)91-1130	ビジネスランド	☎(06) 348-1881	藤井寺店	☎(0729)38-2111	京都近鉄店	☎(075)341-5769	田原本店	☎(07443)3-4041
本厚木店	☎(0462)25-5151	寺地店	☎(0762)47-2524	高槻店	☎(0726)85-1212	岸和田店	☎(0724)37-1021	大久保バイパス店	☎(0774)44-1211	熊本店	☎(096)359-7800

ピュア32bitMC68EC030搭載。
クリエイティブパワーが花開くX68030シリーズ。

X68030

本体+キーボード+マウス・トラックボール
5.25インチFDDタイプ CZ-500C-B(チタンブラック)標準価格398,000円(税別)
HDタイプ CZ-510C-B(チタンブラック)標準価格488,000円(税別)

X68030 Compact

本体+キーボード+マウス
3.5インチFDDタイプ CZ-300C-B(チタンブラック)標準価格388,000円(税別)
HDタイプ CZ-310C-B(チタンブラック)標準価格478,000円(税別)

●写真のカラーディスプレイは別売です。

なか身は、どちらも32ビット。

プロセッサの未来を先取、洗練されたアーキテクチャを誇るMPU MC68000シリーズを搭載。
先駆のクリエイティブ・アビリティで使う人の創造性に応える68ワールドへ、どうぞ。

32bit内部演算処理※16bitバスアーキテクチャ。
潜在能力を秘めたX68000シリーズ。

X68000 XVI

本体+キーボード+マウス・トラックボール
5.25インチFDDタイプ CZ-634C-TN(チタンブラック)標準価格368,000円(税別)

X68000 XVI Compact

本体+キーボード+マウス
3.5インチFDDタイプ CZ-674C-H(グレー)標準価格298,000円(税別)

※X68000シリーズはMC68000(内部レジスタ32ビット、16ビットバス)を搭載しています。●写真のカラーディスプレイおよびカラーディスプレイテレビは別売です。

●お問い合わせは…
シャープ株式会社 コンシューマーセンター西日本相談室〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表) 電子機器事業本部システム機器営業部〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)

T1002179080608 雑誌 02179-8